CURIOSITY DRIVEN EXPLORATION WITH FOCUSED SEMANTIC MAPPING

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAIʻI AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

MECHANICAL ENGINEERING

DEC 2019

By

Curran D. Meek

Thesis Committee:

Zhuoyuan Song, Chairperson
Zachary Trimble
Narayana Santhanam

Keywords: Hybrid Focus, Robotics, Exploration, Curiosity, Semantics, Topic Modeling

ACKNOWLEDGMENTS

First and foremost I would like to thank Dr. Song for always providing new challenges and exemplary assistance in my learning and pursuit of this thesis. I would also like to thank Dr. Santhanam for always answering my many questions outside of class, enabling me to digest and understand many concepts. Furthermore, I thank Dr. Berkelman for inspiring me to strive for excellence in robotics. Finally, I want to thank all the selfless individuals who post tutorials, guides, thoroughly answering questions, and troubleshooting information online. Without the assistance of the internet I would have never accomplished this thesis.

ABSTRACT

Among many similar natural creatures, humans have the impressive ability to seamlessly explore a new environment and build up their knowledge base. By combining complex exteroceptive perceptions we can "easily" identify the terrain, objects, or hazards around us allowing us to determine a target of attention to make behavioral decisions accordingly. However, sophisticated exploration behaviors are still nontrivial to today's mobile robots, despite the recent progress in sensing and processing capabilities. Generally, robots face many challenges in achieving human-level exploration, including sufficient perception for navigation, intelligent knowledge extraction and representation, and autonomous decision-making that maximizes both perception and learning. Semantic description is a promising technique to assist mobile robots to sparsely represent and accumulate knowledge of their environments. In this project, we propose a hybrid focus model scheme using curiosity-driven exploration coupled with a prior knowledge base. In this scheme, the robot uses realtime topic modeling to discriminate the sensing data, with a one-stage feature detector to identify the most curious feature in the scene. Using this topic modeling based curiosity, focus driven adaptive sampling and navigation are implemented to facilitate more robust decisions and maximize the perception within the focused region. The proposed hybrid focus combined with metric information allows the creation of semantic maps that enhance human and robot interaction. Using both underwater video datasets and data collected by a mobile robot platform with a RGB camera in a laboratory setting, the performance of the proposed approach is demonstrated and analyzed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ROV	Remotely Operated Underwater Vehicles
AUV	Autonomous Underwater Vehicles
CNN	Convolutional Neural Network
YOLO	You Only Look Once (algorithm)
LDA	Latent Dirichlet Allocation
ROST	Real-time Online Spatiotemporal Topic Modeling
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
ORB	Oriented Brief
ROS	Robot Operating System
LiDAR	Light Detection and Ranging
RGB	Red, Green, Blue
RVIZ	ROS Visualization
pLSA	Probabilistic Latent Semantic Analysis
PPX	Perplexity
ROI	Region of Interest
INS	Inertial Navigation System

CHAPTER 1
INTRODUCTION

1.1 Motivation

Humans have long aspired to explore the ocean depths, but the many challenges of the ocean have required innovative thinking. The hazards presented in ocean exploration have required the command of technology to reach the mysterious depths. Using an autonomous robotic system is one such technological solution for the task. These robots can be used for a multitude of applications such as understanding sea life, examining the state of coral reefs, or even as early warning detectors for national security. Often scientists utilize remotely operated underwater vehicles (ROV) or autonomous underwater vehicles (AUV) to explore specific areas of the ocean. These robots have sensors to monitor various conditions but are employed by using a pre-planned path. This path is non-trivial, as decided by the scientist, but since the robot has a limited decision capacity it can easily miss the very thing it is sent to observe. For instance, as Flaspohler mentions in [1], the underwater robot used in an expedition captured some very interesting images of a crab migration, but since it was preforming a pre-planned path it did not follow the crabs, continuing to observe subjectively uninteresting rocks and sand. Here is just one application where human-like intelligence can greatly help underwater endeavors, an example case can be seen in Figure 1.1.

Underwater robots use a wide variety of sensors to help gain sufficient information about the environment. One widely adopted device is the camera for the rich information contained in their measurements. Modern artificial intelligence approaches can allow cameras to be used for a semantic understanding of an environment, but the underwater environment poses increased challenges since the image quality is often highly susceptible to the lighting condition, water quality, and even depth [3]. Instead of explicitly learning various objects

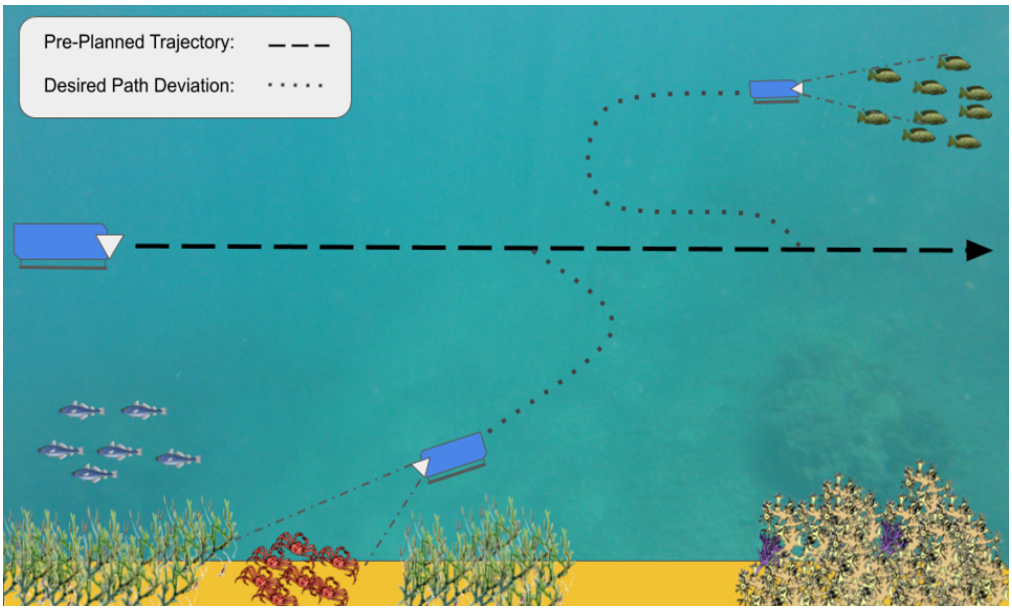

Figure 1.1: This image demonstrates autonomous decision making based on detecting curious objects. Instead of simply following a pre-planned path, the robot is able to notice interesting things and change the navigation plan to better observe them [2].

from a large dataset, which could be skewed by the image quality, topic modeling has been proven successful in generating a semantic understanding of the environment that does not rely on specific training data. This semantic description is a promising technique to assist mobile robots to represent and accumulate knowledge of their environments in an efficient yet traceable fashion [4].

So, how do you program curiosity into a robot? A variety of methods have been used, [5] or [6], but since our own understanding of curiosity is incomplete, no single method is considered optimal. Curiosity, with regard to exploration, can be understood as taking information gaining actions in order to reduce one's uncertainty [7]. This can be motivated through *external*, motivation to accomplish a goal, or *internal*, inherent desire for gaining specific information [7].

Modeling externally motivated curiosity is straightforward and can often be modeled using a cost, or reward function, in conjunction with machine learning techniques [8]. In

the field of underwater robotics a topic modeling framework has been used as in [9], with improvements made in [10], that created a very robust, promising methodology for internal curiosity driven exploration. The exploration system uses topic modeling that segments images based on visual words. These visual words are low-level features derived from the various color and saturation of the different pixels. Essentially the robot attempts to learn the distribution of these visual words in its environment. By comparing the current distribution of topics, groups of similar visual words, the robot gains an understanding of common occurrences within the environment. The robot can then understand something as new or curious by comparing a new topic to neighboring observations. Thus, a region of a high curiosity creates a goal or point for the robot to investigate or explore. This methodology is the foundation for the goal of this paper. We aim to demonstrate how to provide a robot with a robust focus point that can be used to direct robot motion. To improve upon this existing method we have introduced several new concepts, namely a threshold placed upon the topic perplexity and a hybrid focus model.

Figure 1.2 introduces the logic for our hybrid focus model. The logic flowchart shows how the overall implementation that would be used for underwater exploration. Taking inspiration from focus concepts, a region of interest can be determined and a change in navigation can be made. This work focuses on the lighter shaded boxes to fully implement the combination of focus modes.

The following description describes Figure 1.2 in its entirety. Although as mentioned, the work focuses on the lighter color boxes. The objective of the hybrid model is to determine the most curious region within the image, based on topic perplexity, but different from known objects. Topic perplexity is found with topic modeling. Topic modeling is an unsupervised method for semantically discriminating an image based on visual words. An unsupervised learning method removes the need for prior training lending itself well to exploration. To start, the hybrid focus model will obtain topic distribution of the image. This distribution

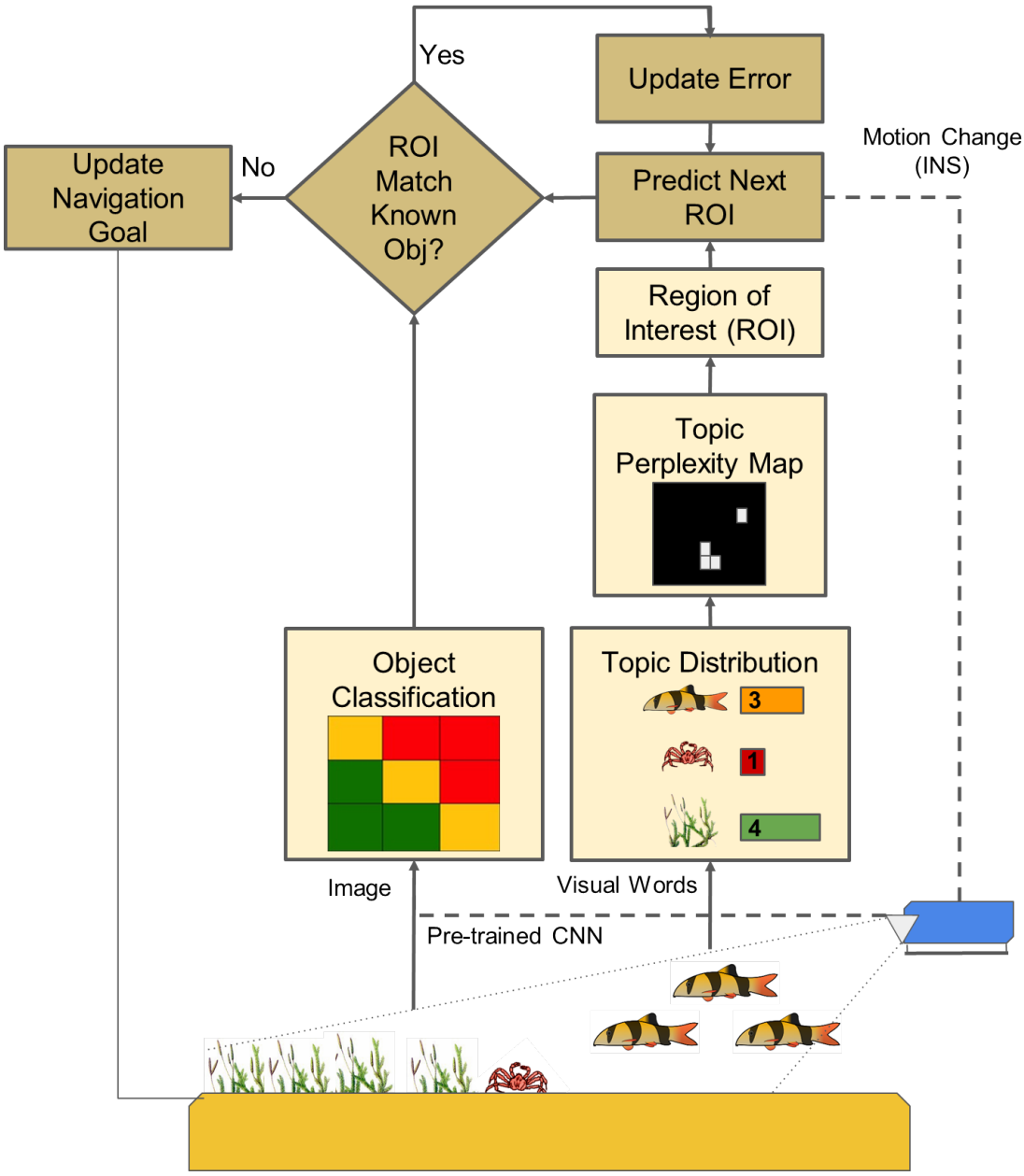

Figure 1.2: Topic modeling is used to determine the region of interest (ROI) based on the visual words present. Visual words are groups of low-level features (contrast, orientation, color) that are grouped into topics. The topics are used to generate perplexity, based on inverse probability, that are then mapped onto the image. Perplexity values are used to locate the curious region. Once the ROI is determined, a prediction is made based on the robot's movement, measured by an onboard inertial navigation system (INS). This ROI is analyzed by an object classifier to determine the presence of a known object. The predicted ROI is compared to the object classification to either refine the prediction model or update the navigation goal [2].

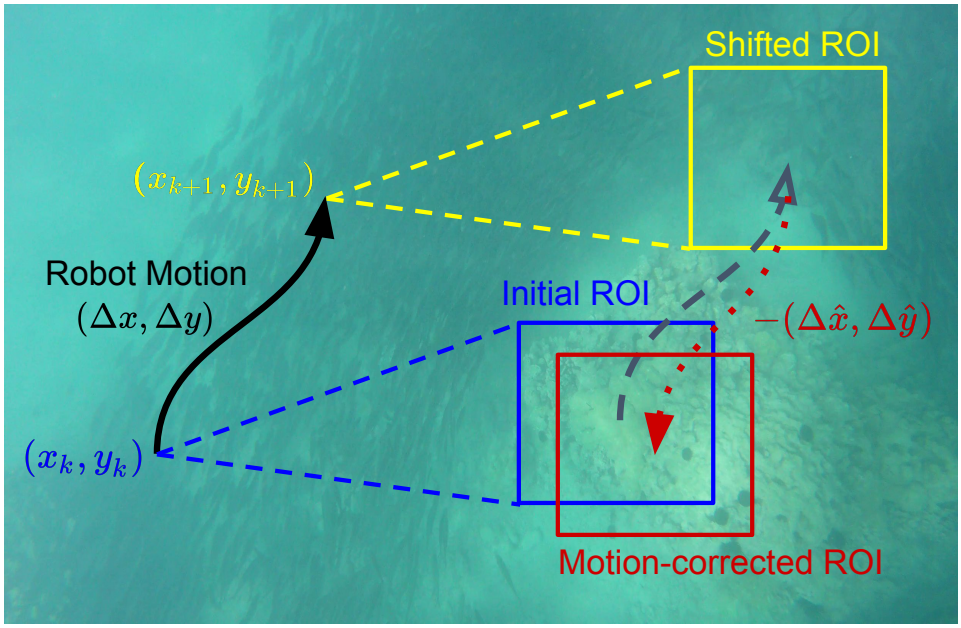

Figure 1.3: The region of interest (ROI) (blue box) shifts due to the robot motion. The motion-corrected ROI (red box) can be obtained by back-propagating the shifted ROI (yellow box) using estimates of motion changes, $(\Delta\hat{x}, \Delta\hat{y})$, based on the robot's navigation system. This concept would help in maintaining the focus region during the robot's movement [2].

uses a down-sampled image to semantically discriminate the environment based on low level features. Using this topic distribution, a perplexity value can be given to objects in the image [10]. The various perplexity values are determined by comparing the current observation to its spatial and temporal neighbors [10]. A point of focus is determined based on the largest topic perplexity. The goal of the robot is to then predict where this curious point will be in the next time-step. This concept is illustrated in Figure 1.3, where the red box shows the robots predicted curious region based on its own movement [2]. This region is then analyzed with an object detection CNN to ensure the region contains an unknown object. Any error between the prediction and known objects can be used to either emphasizing the prediction or object classification for navigation decisions. Utilizing this methodology provides a minimal semantic representation of the environment, facilitating robotic exploration and decision making [2].

The semantic information gained from the hybrid focus model can be combined with a range sensor to allow objects to be correlated with range data for mapping. This map would provide the user with a detailed and useful understanding of the area observed by the robot. The map can be used to track the robots progress, or to update exploration goals for robot control. When combined with other sensor information, the hybrid focus model can provide a robust decision model for underwater exploration.

1.2 Focus Background

Human attention allows incoming information to be restricted, allowing only small important information to reach our memory. This allows the image we see to be processed sequentially [11]. Localized analysis of the small salient features of the image are processed sequentially, instead of analyzing the entire image at once [11]. In addition, attention can be understood by two types top-down and bottom-up [11].

1.2.1 Bottom-Up Focus

As described by [11], *Stimuli induced attention* describes how our gaze is directed to interesting objects based on their contrast to the environment. This attention mechanism can be captured through a bottom-up approach. The bottom-up approach uses low level feature extraction to generate feature maps that explain the most salient objects in an image. Initially, the low-level features such as colors, intensities, or orientations are extracted in parallel. These are then used to create feature maps that describe the spatial contrast of each feature in the image. These maps are then combined to generate a unique salience map that topographically encodes the magnitude of salience for each pixel of the image. This map can be processed to find the location of the highest saliency point, ultimately outputting the spatial location of the object of interest [11].

1.2.2 Top-Down Focus

The previous work [11] further categorizes focus with *directed focus*, or referred to as top-down focus, as the second form of attention. This type of attention is a directed task, such as looking for the fish in a picture. This more powerful form of attention uses higher order thought processes of the brain creating a top-down approach. The eyes will be directed to a specific location by the top-down focus based on the task. Humans can leverage their large knowledge base of the environment to more readily direct their focus. In humans, this top-down focus is operated in parallel with the stimuli induced focus [11]. Top-down focus is more computationally expensive than the stimuli induced counterpart requiring "200ms or more" to process in the human brain [11]. A computational method for implementing a combined focus model has been studied in robotic applications in [5]. This model emulates a top-down focus mechanism to create a feedback loop for control of teleoperated robots.

Memory networks are another method for implementing attention mechanisms computationally. "In neural networks, attention primarily serves as a memory-access mechanism" [12]. These networks are a form of short-term memory that vary the weights of input features. As preformed ina recent work [13], the network learns what inputs are more valuable to use in making predictions. For example, a neural network model identifies the most salient points in an image, like that of the bottom-up approach, then generates a word to best describe that portion of the image. The network then iterates through the next most salient point and again generates a word to describe it. This continues until a descriptive caption is generated for the image [13].

The memory network approach contrasts to our proposed hybrid focus methodology. Instead of simply iterating through salient points, the salient points are also analyzed for previous understanding using an object classification algorithm. Our hybrid focus model attempts to create a parallel focus model similar to [5] but combine the information differently. We aim to use the information to direct the robots motion, vice predict the motion of

another object. Using both types of focus we can extract the location of the most interesting point in the environment, providing a clear target for the robot to observe.

1.3 Computational Methods Background

1.3.1 Object Classification

To better understand the need for employing a topic modeling framework it is important to examine current computer vision work. Computer vision has been a large area of research in the computer science community. Many techniques have been developed to tackle very difficult challenges in object detection and semantic representation. These techniques utilize convolutional neural networks (CNN) that can identify, classify, and track multiple objects in real time, as seen in [14]. CNN pose several challenges for use in underwater robotics. Firstly, these networks are trained on pre-labeled images that the network can learn to recognize the various objects. These training datasets need to be large and varied in order for the network to generalize well to new data. In the underwater environment, not only are the available images limited, but the image quality is highly varied based on the water conditions, light available, or depth when the image was taken [15]. Alternate approaches can be taken, such as using a bag of words model. This model, similar to topic modeling, segments the image based low-level features. Thus simplifying what the network needs to learn for correct identification of an object, allowing for a more efficient method that can be used on a typical cell phone [16]. Unfortunately, prior knowledge or data is not often available for unexplored areas. Since we do not even know what to expect ourselves, training a robot to learn the unexpected proves difficult.

Despite the negative aspects of the traditional computer vision approach, it still provides very useful information in real-time. For instance, the HyperNet RPN (Region Proposal Network) achieves real-time speed, 5 frames per second (fps), with high accuracy out preforming

its successors (Fast R-CNN, and Faster R-CNN) [17]. While simpler methods, such as YOLO (You Only Look Once), achieve real-time processing speed at the sacrifice of classification accuracy. Either method can recognize a multitude of objects depending on the dataset used. The COCO dataset provides nearly 80 objects that can be recognized [18]. Fortunately, with the development of small, high power processing platforms with graphics processing units (GPUs) such as the Nvidia Jetson Nano, these realtime processes can be used on small platforms. Even though these algorithms require prior training, they are still important to be used in underwater exploration since they allow the robot to have a knowledge base to compare against the surrounding environment. Because of these reasons, we employ the YOLO object recognition scheme as the top-down portion of our focus methodology.

1.3.2 Topic Modeling

Latent Dirichlet Allocation (LDA) is a hugely successful form of topic modeling that is widely used [19]. Originally purposed for text processing, this form of topic modeling assumes that there are inherent topics randomly mixed within a corpus. These latent topics describe an overarching specific distribution of words that are related. LDA finds more semantically relevant topics by placing a Dirichlet distribution prior on the topic distribution and word distribution, which prompts them to be sparse [19]. Furthermore, topic modeling has been re-purposed for computer vision applications, such as [20] or [21].

Object discovery refers to the ability to find new objects within the environment without any prior knowledge of them. In the context of topic modeling, an unsupervised approach would aim to find a dominate object class, and localize the object given a series of images. The most basic approach is to preform LDA topic modeling, but this method can generate topics without any clear meaning, known as the coherence issue. More specifically, the previous work [22] expresses that the perspective words that describe a topic are incoherent with each other. The LDA model can be improved upon by adding must-links. Must-

links provide constraints on how given words relate to each other and the topic that they describe. Generally, the links constrain all topics within the corpus. To improve even more, these must-links are generated with prior semantic knowledge, only constraining one or some of the topics within the corpus. This method is employed to LDA with mixed Dirichlet trees (LDA-MDT). Using a bag of visual words, that are created by clustering local features in the image, the topics are extracted with the LDA-MDT method, overall resulting in improved topic coherence is improved which facilitates object discovery [22].

Alternatively, to adequately preform topic modeling for robots, the approach must be modified. In order for a robot to use the topic model information, it must be updated continuously, since robots experience time and space changes continuously. The topic model must be refined to account for new and old observations. These challenges resulted in the development of the realtime online spatiotemporal topic-model (ROST) [9]. Initially a vocabulary of visual words is generated by using scale-invariant feature transform (SIFT) [23], speeded up robust features (SURF) [24], or oriented brief (ORB) [25]. This vocabulary is generated using an unrelated dataset from what the robot would encounter [9]. These features provide groupings of visually related features that form visual words. Visual words are extracted from the robots observations and compared to the known vocabulary. LDA is then utilized to form the topic distribution over the image. ROST relaxes the assumption made in LDA, that each image would have an independent topic distribution [9]. This allows the topics to depend on spatial and temporal neighbors [9]. The topics are then refined using a realtime Gibbs sampler to compare the incoming observations to previous observations as the robot travels through the environment [10].

The newly gained topic information can be used in a variety of ways. Originally, in [9] a summary of the topics for various locations was saved and a surprise score was generated. Based on the deviation of the current observation from the summaries, the program was allowed to emulate curiosity. Although, this method proved useful, it was later refined to

base the curiosity on weighted factors, such as word perplexity or topic perplexity [10]. Perplexity in this context means the inverse probability of the visual word belonging to a known word or topic. This provided the framework for our curiosity driven methodology and bottom-up component.

1.4 Core Contributions

This work has introduced two main new concepts that build upon existing work.

1. The hybrid focus model introduced is a novel computational method for combining different types of focus. Both the top-down and bottom-up focus types are combined to generate a comprehensive point of focus.

2. Building upon the ROST methodology, a grouping and thresholding methodology is applied to the perplexity values. The grouping and thresholding provide a simplified representation of the topic perplexity generated by ROST.

In addition, a conference paper [2] and poster, seen in Appendix A, have been published in the MTS/IEEE OCEANS Conference in Seattle, WA in 2019 in pursuit of this work. This paper was selected as one of the twenty finalists for the Office of Naval Research Student Poster Competition.

1.5 Thesis Outline

This thesis details the development of a hybrid focus scheme. The hybrid focus scheme takes advantage of the ROST framework for the bottom-up component and YOLO as the top-down component. This focus scheme was developed using the robot operating system (ROS), Python, and C++. Testing was preformed on underwater video and a TurtleBot3 burger, which is an indoor land based robot.

Chapter 2 discusses the methodology and implementation of the top-down focus. This component of the model is developed with the YOLO algorithm. The focus is also incorporated with a semantic mapping algorithm deployed on an indoor robot that makes use of ROS. A description of the YOLO algorithm and ROS architecture are included.

Chapter 3 demonstrates the use of the ROST Interpreter for the bottom-up focus, using python and underwater images. LDA topic modeling is explained including its use in the ROST framework. The methodology of the ROST Interpreter program and testing with images is detailed.

Chapter 4 discusses the development of the hybrid focus model. The chapter presents how the algorithm combines the focus frameworks, along with how it was tested on underwater video and an indoor robot. Indoor testing builds on the semantic mapping concept by adding new object labels.

Chapter 5 summarizes the work and provides ideas for future work.

CHAPTER 2
TOP DOWN FOCUS AND SEMANTIC MAPPING

2.1 Introduction

The first challenge in the hybrid focus development was to understand and test how to relate a top-down focus mechanism with a metric map. Our initial goal was to locate objects of interest within an environment and produce a map with semantic labels detailing where that object was positioned. Using semantics help bridge the gap between how humans and robots understand the world. Robots need to be able to determine high level relations of objects in the environment rather than simply spatial relations [26]. Generally robots understand the world with metric information. A LiDAR tells the robot its distance from given objects, or a wheel encoder gives how many revolutions have occurred. These numbers are not intuitively easy for a human to understand. For instance, it is much simpler to tell a robot to go to the kitchen, than to find out the coordinates or distances associated with the kitchen's position.

As discussed in Chapter 1, top-down focus is like a directed task (e.g., "find the red ball"). This type of focus requires a prior knowledge base and semantic understanding of objects. To implement the top-down focus we used the you only look once (YOLO) object recognition algorithm. The results from YOLO produce a bounding box that locates a known object within the scene. This information can be combined with metric information to plot its location within a map.

2.1.1 YOLO Description

The YOLO algorithm, [14], is able to preform realtime object detection by running image classification in parallel with object location. This unification of components in a single neural network simplify traditional CNN image classifiers. The algorithm first processes the

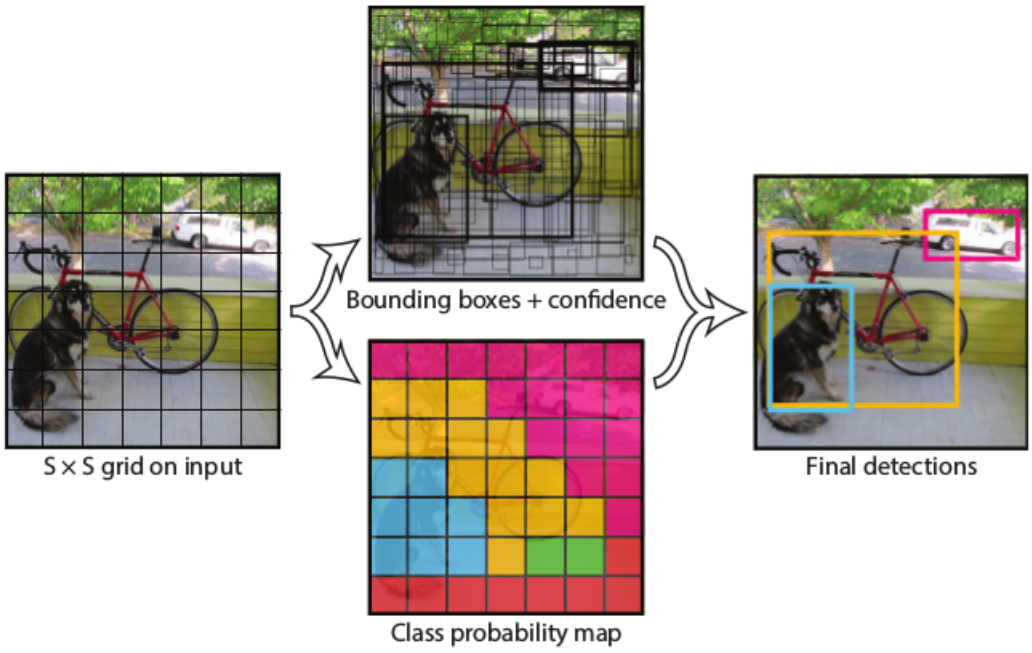

Figure 2.1: This figure shows a simplified process of how the YOLO network detects objects. The image is segmented into a grid that is used for two parallel processes: bounding box predictions, and class predictions. These predictions are filtered to leave only the highest probability objects. [14]

entire image by segmenting it into a grid. Each cell then predicts a fixed amount of bounding boxes and confidence scores for each box. The confidence, d, is defined as

$$d = \mathrm{P}(\mathrm{Object}) * IOU_{\mathrm{pred}}^{\mathrm{truth}}$$

where $IOU_{\mathrm{pred}}^{\mathrm{truth}}$ is the intersection over union of the predicted box and ground truth. In parallel, each grid cell predicts the class within the cell containing an object. Only one class is predicted in the cell regardless of the bounding boxes it contains. Finally, the predictions are filtered out to only leave the highest scores of detected objects. Figure 2.1 visually represents this process [14].

2.1.2 Robot Operating System

ROS is an open-source software for operating systems on a robot. As detailed in [27], many tools and libraries are provided to interface with various hardware and programming languages. ROS uses a peer-to-peer network framework to form a publisher and subscriber concept. Publishers and subscribers are used to either send information or retrieve information within nodes. Nodes are programs that preform a specific task. For instance, one node can be used to publish images from a camera. Other nodes can then subscribe to the camera images for their own use. ROS can be used for a variety of purposes facilitating a simple way for robot control [27].

2.2 Algorithm Methodology - Mapping

We created a system that could combine metric and semantic information for the use of robot exploration. Specifically, the robot was tasked to autonomously identify and localize objects of interest specified beforehand. The algorithm was implemented in ROS and programmed using Python. Ideally, the algorithm could be used with various types of sensors that are sufficient for metric-based simultaneous localization and mapping (SLAM) and semantic information extraction, but for this implementation a RGB camera and LDS-01 LiDAR was used.

To gather semantic information a ROS package, *darknet ROS* [28], was used. This program made use of the YOLO object detection algorithm [14]. The YOLO framework was used over other CNN because it achieves realtime object detection. It was trained on the COCO dataset [18] that contained information on approximately 80 different objects. The darknet ROS program was able to processes incoming images and publish the bounding box information. The bounding box coordinates were used for correlating the detected object with the map [2].

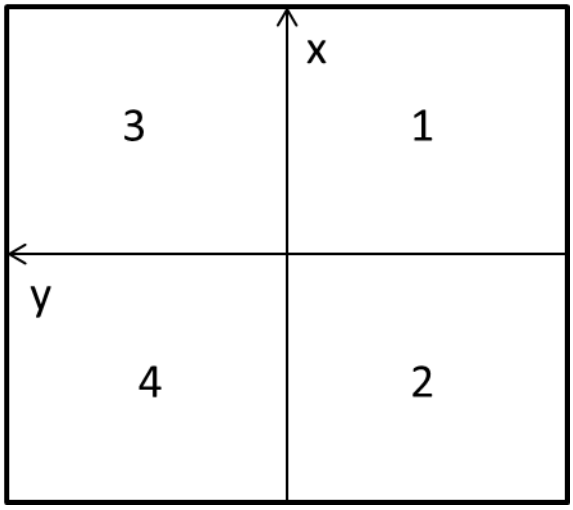

Figure 2.2: The figure shows the positions of the x-axis and y-axis in the map frame along with defined quadrants for determining relative positions of the robot and object of interest.

The metric map information was gained from OpenSLAM's *Gmapping* package. The Gmapping package combines ranging information from the LiDAR and position information from the wheel encoders to generate an occupancy grid map to preform SLAM [2]. An occupancy grid map is generated that represents the robot's understanding of the world around it. Occupancy grid maps contain three states, either empty, occupied, or unexplored. Objects would be represented as an occupied area in the map.

The information from the object detection and SLAM components were combined to create a map with the marked location of an object of interest. First, after the image has processed through YOLO, when the results contain the label of the object of interest, the center point of the bounding box is determined. This position is then correlated to the corresponding heading angle of the LiDAR based on the field of view of the camera. The heading angle is calculated based on the orientation of the robot and the relative position of the object. Figure 2.2 shows the coordinate and quadrant layout used for calculating the object of interests' position with respect to the map frame. In ROS the x-axis and y-axis are switched, while the quadrants are self defined for use in the algorithm.

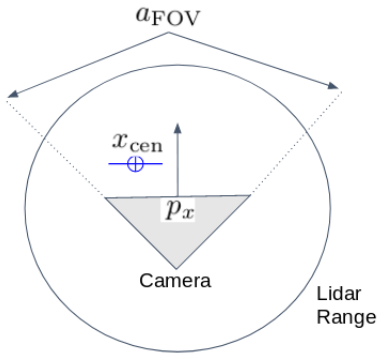

Figure 2.3: The figure is used to illustrate how the pixel position in the camera is correlated to the LiDAR range. Here the circle represents the LiDAR coverage, $0° - 360°$. The triangle represents the camera, with the field of view, a_{FOV}, indicated and center point designated with the arrow. The p_x represents the pixels in the x direction, while the x_{cen} represents the center pixel of a detected object of interest. The Equations (2.1) and (2.2) are used to determine the correlating heading angle.

First, the orientation of the robot and correlated position of the object is determined. To accomplish this, the pixel position of the object is converted to an angle that is then used to correlate to the angle of a LiDAR sensor. So the relative position of the object requaires the object heading angle, h, to be determined differently. If the object is left of center, the following equation is used

$$h = \frac{a_{\text{FOV}}}{2} - (x_{\text{cen}} * \frac{a_{\text{FOV}}}{p_x}) \tag{2.1}$$

or if the object is to the right of center, h is calculated by

$$h = 360 - \frac{a_{\text{FOV}}}{2} - (x_{\text{cen}} * \frac{a_{\text{FOV}}}{p_x}) \tag{2.2}$$

where x_{cen} is the center x coordinate of the object bounding box, a_{FOV} is the max field of view angle for the camera, and p_x is the total horizontal pixels of the image. This correlation of a camera and LiDAR can be seen in Figure 2.3. The heading angle is rounded, and used to retrieve the range reading of the LiDAR at that angle [2].

The quadrant of the object is then determined using a comparison of the robot's angle and the object's heading angle

$$a_{\text{check}} = \psi_r * (180/\pi) + h$$

where ψ_r is the yaw angle of the robot. Using a series of if statements, the a_{check} angle is used to determine if the robot and object are oriented in the same quadrant.

Based on the orientation of the robot and determined quadrant of the object of interest, the position of the object is calculated. The position of the robot is either added to or subtracted from to retrieve the coordinates of the object. Algorithm 1 shows an example logic flow in determining how the object distance is either added or subtracted. Then using the range to the object, robot position information, and heading angle, the position of the object of interest, in the map frame, is calculated by

$$x_m = \pm r * \sin(\alpha) + x_r$$

$$y_m = \pm r * \cos(\alpha) + y_r$$

where (x_m, y_m) represents the position of the object, (x_r, y_r) are the position of the robot, r is the range from the robot to the object, and α is the relative angle between them. Angle α is based on the orientation of the robot and the heading of the object. It is calculated by either $\alpha = \psi + h$ if both robot orientation and object are in the same quadrant, or $\alpha = \psi + h$ if the robot and object are in varying quadrants.

Algorithm 1: Object of Interest Position Calculation

```
Initialization: h (heading from bounding box center), r (ranges from LiDAR), (x_r,
  y_r, ψ_r)(robot position);
while object of interest detected do
    if Robot is pointing to quadrant 1 and object is on the Right then
        Check to see if object is in the same or different quadrant, a_check
        if If a_check > 90 then
            Subtract x and y range component from both x_r, y_r
        else
            Add x range component to x_r
            Subtract y range component from y_r
        end
    else
        Repeat for all other quadrant combinations
        ⋮
    end
end
```

This algorithm simply determines the quadrant orientation of the robot relative to the map frame by calculating the x and y position of the object based on the quadrant it is located from the robot. The quadrants are as shown in Figure 2.2. A series of *if* statements are used to ensure the object position is correctly added or subtracted from the robot's location.

Once the position information has been calculated, a marker is then published to RVIZ. RVIZ is a graphical user interface within ROS that allows the user to visualize all the information about the robot. Since the robot position, LiDAR returns, and the occupancy

grid map are all shown in RVIZ, it is the best place to semantically represent what the robot has seen. In this case a marker object is published to RVIZ that creates a yellow circle at the position calculated [29].

2.3 Testing

The goal for this test was to ascertain if the algorithm discussed properly correlated the image and LiDAR data for an object of interest.

2.3.1 Simulation

Testing was first preformed using simulations with Gazebo ROS. The TurtleBot3 was simulated in a simple enclosed environment with nine cylinder obstacles. The simulation provided all sensor readings from the TurtleBot3 and RVIZ integration.

To test the ability to create markers and understand the coordinate system in ROS, the position of the interesting object was pre-programmed at a fixed angle. As the robot was virtually teleoporated, it marked objects at the fixed angle. The results of this can be seen in Figure 2.4.

Having the ability to correctly place markers within the map, a webcam was then integrated to test the functionality of correlating an object of interest. An object of interest, when observed by the webcam, was used to trigger placing a marker onto the map. The robot was positioned in the same simulated environment, while the object of interest angle was fixed. So, as the object of interest (a person) was put into view of the camera, the marker was immediately placed at the fixed angle location, as shown in Figure 2.5.

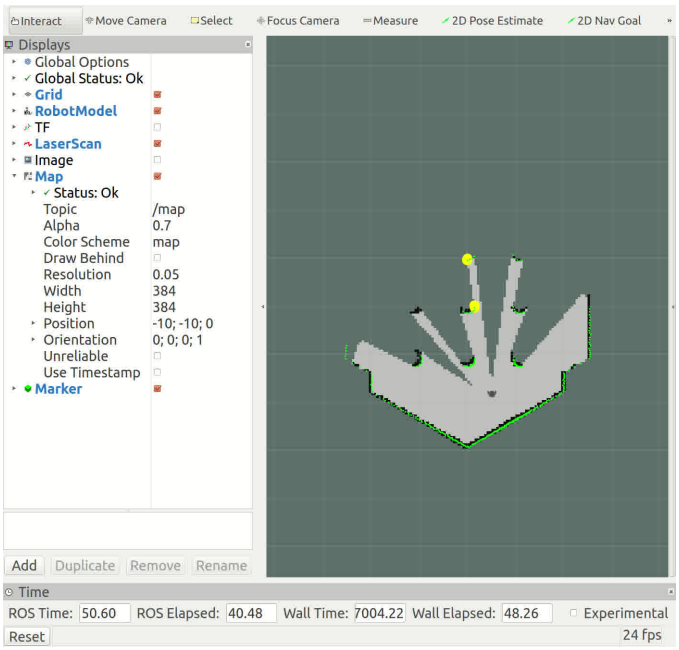

Figure 2.4: The yellow circles in this image represent the object of interest. In this testing case the heading of the object of interest was programmed as a fixed angle. Objects at the fixed angle were continually marked with a yellow circle as the robot moved through the environment.

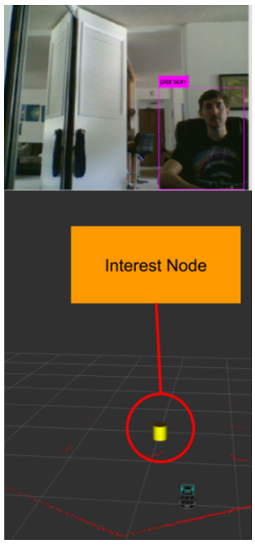

Figure 2.5: The simulated TurtleBot3 correlates the webcam image with the virtual environment to map the object of interest. In this case the object of interest was set as a person. The yellow cylinder shows the marked location of the object of interest when triggered by seeing the person.

2.3.2 Indoor Testing

TurtleBot3 Setup

The algorithm was then implemented with the TurtleBot3 Burger in an indoor environment. The TurtleBot3 Burger is equipped with a LiDAR LDS-01, two DYNAMIXEL servos, and controlled with a Raspberry Pi 3. A Raspberry Pi camera module v2 was mounted to the front center-line and connected to the Raspberry Pi 3 via a ribbon cable. The Raspberry Pi 3 runs ROS Kinetic in order to communicate with the various sensors and transmit data over Wi-Fi. Several pre-built ROS packages were utilized, such as the raspicam node to retrieve images from the camera module. The rest of the pre-built packages for the TurtleBot3 were setup as directed in [30]. The nodes created for this testing have been described, but the high level interaction with the other nodes can be seen in Figure 2.6.

Environment Setup

The environment was a living room area with three wine glasses, a bottle, and a sofa present. The TurtleBot3 was teleoporated to explore the area and generate the occupancy grid map. Once the objects were mapped as occupied, the TurtleBot3 was then positioned to view all the objects. The algorithm was able to detect each object within the environment and properly label them. In this scenario, the bottle was considered the object of interest. As such, a marker was placed onto the occupancy grid map at the bottle's location, shown in Figure 2.7.

Performance

After several iterations of testing, the accuracy of the marker placement was determined. The marker accuracy was partly dependent upon the LiDAR accuracy, as listed in Table 2.1. Errors also arise from rounding in the pixel to angle conversion, Equations (2.1) and

(2.2). These errors can cause an object to be mislabeled within the occupancy grid map. Furthermore, in this test the image from the robot was transmitted over Wi-Fi to then be processed on a laptop. This resulted in very slow processing, approximately 0.1 frames per second (fps), which limited the response time of the robot.

Table 2.1: Performance of the general semantic approach [2]

Image Classification Speed	0.1–0.2 fps
Average Classification Probability	70%
Marker Range Accuracy	±19.5 mm
Marker Range Precision*	±13 mm
*Based on LiDAR (LDS-01) specifications	

2.4 Conclusion

In this chapter we have described the top-down focus development. The top-down focus was adequately demonstrated using the YOLO algorithm. The robot was able to consistently identify the trained upon objects. Furthermore, the object location as given by YOLO, was able to be correctly correlated with sensor information to mark the desired location of a given object. By combining the camera and LiDAR information, we were able to generate a map with semantic information that can aid in human robot interactions.

With the top-down focus and semantic mapping demonstrated we moved forward to the bottom-up focus. The following chapter will cover the background information needed to develop the bottom-up focus, in addition to how the concept was implemented. The methodology was tested utilizing underwater images, with the results being discussed.

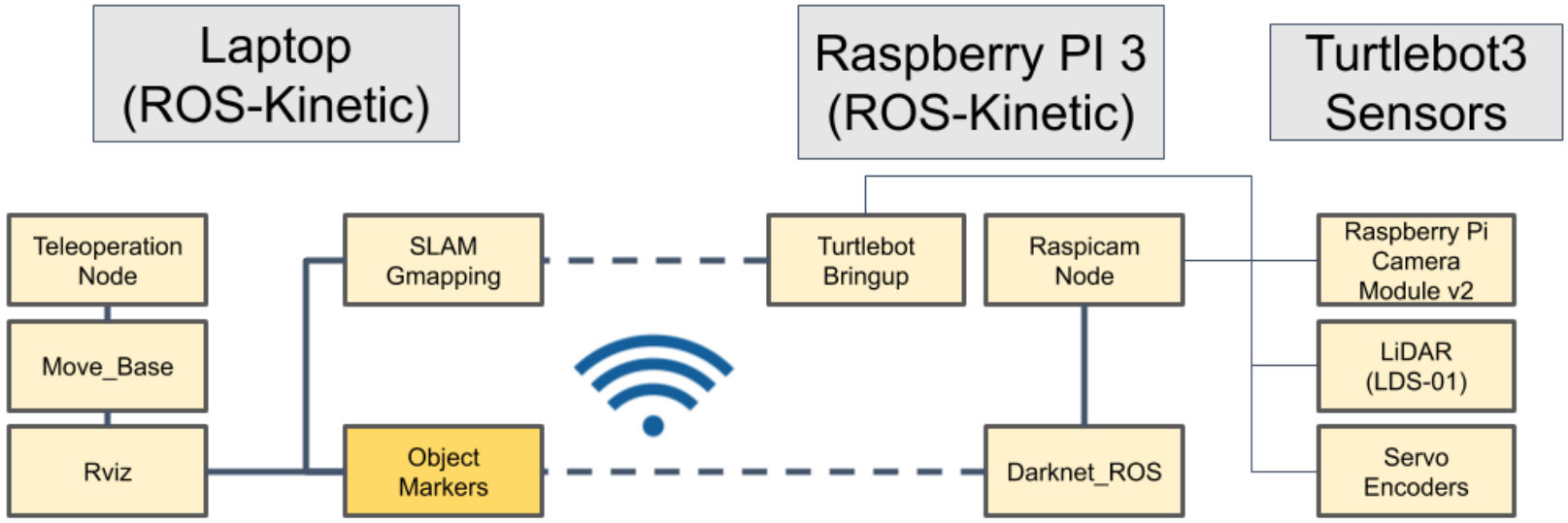

Figure 2.6: Every box represents a separate ROS node used in the testing of the top-down correlation. The light yellow boxes represent the nodes and programs created for this work, while the tan boxes are pre-built packages in ROS. The dashed lines represent communication over a wireless network, while the solid lines represent directly connected communication. More information about the pre-built packages can be found at [30] and [27], or the created nodes at `https://github.com/song-ranlab/curisoity_exploration_and_hybrid_focus`.

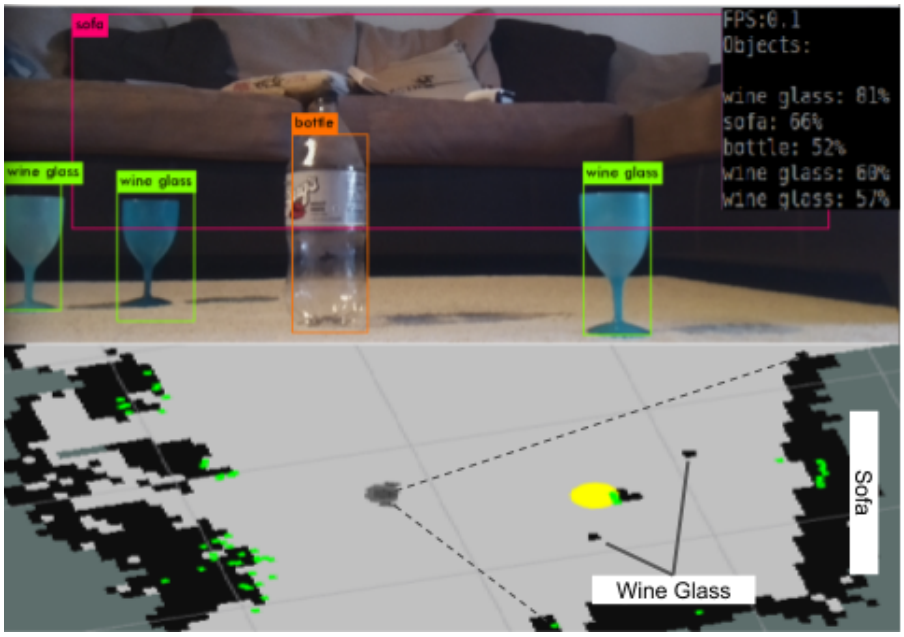

Figure 2.7: The top of the figure shows the image view as seen through the robot's perspective. The objects are labeled and bounded within the image. Object classification confidence for each object is shown in the top right corner. The bottom portion of the figure shows the corresponding occupancy grid map of the scene, with a yellow marker marking the object of interest, a bottle. Because of the small size of the wine glasses, only two gave consistent LiDAR returns [2].

CHAPTER 3
BOTTOM-UP FOCUS

3.1 Introduction

The previous chapter detailed how the top-down focus was used. In this chapter we aim to detail the background understanding, methodology, and testing in the development of the bottom-up focus. Here the bottom-up component serves as the main driver for robot attention. It relies on topic modeling of the environment for semantic understanding. The low-level features observed are extrapolated into high-level topics that are used to determine the curiosity of a region. Here we describe how topic modeling is preformed and how the ROST Interpreter algorithm was developed.

3.1.1 LDA

To expand upon the topic modeling ideas, as discussed in Section 1.3.2, it is important to note the details of LDA and statistics. This section is intended to give a high-level overview of how the topic modeling equations were developed and used. Firstly, conditional probability should be understood. Conditional probability is the probability that an event occurs given some knowledge about the outcome of the event [31]. This is typically expressed as $P(A|B)$, where A is the event being predicted and B is the knowledge known or assumed about the outcome of event A [31]. This concept is furthered by the use of Bayes' Theorem, given by

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

To put simply, Bayes' Theorem essentially describes that the probability of an event can be calculated using other known probabilities [32]. These two topics form the foundation

"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

Figure 3.1: Example of topic modeling using LDA. The top list of words represent the distribution of words used for that topic. The bottom box is an example text with color coded words that represent the topic from the above list they belong to.[19]

used when creating the LDA equations.

Using the ideas of Bayes' Theorem and conditional probability, the previous work [19] is able to create a probabilistic framework to determine topics within a document. LDA builds upon the initial idea from [33] that there are latent topics that are distributed within a document. The probabilistic latent semantic analysis (pLSA) is able to give a semantic description of a document but is susceptible to overfitting to a dataset during training. Without re-deriving the equation, the LDA approach mitigates this issue by placing Dirichlet priors, α, β, onto the word distribution, ϕ, and topic distribution, Θ, given by

$$p(\mathbf{w}|\alpha,\beta) = \int p(\Theta|\alpha)\left(\prod_{n=1}^{N}\sum_{z_n} p(z_n|\Theta)p(w_n|z_n,\beta)\right)d\Theta \tag{3.1}$$

where $\mathbf{w}$ is the set of words, z_n is the set of N topics distributed over a topic mixture Θ, and w_n is a specific word within the document [19]. The Dirichlet prior α is a vector of values assumed *a priori* on the topic density within a document, and the prior β is a vector that represents the word density within a topic [19]. This equation serves as the model for representing topics within a document. Figure 3.1 is a text example of processing a document using equation 3.1. The document in the figure has a distribution of four topics, where each topic consists of a distribution of similar words.

Applying the Dirichlet distribution results in a sparse topic mixture, which lends well to finding more semantically relevant topics. A Dirichlet distribution is a continuous multivariate probability distribution [34], given by

$$f(x_1,\ldots,x_{\mathrm{K}};\alpha_1,\ldots,\alpha_{\mathrm{K}}) = \frac{1}{\mathrm{B}(\alpha)}\prod_{i=1}^{\mathrm{K}} x_i^{\alpha_i-1}$$

where $(x_k)_{k=1}^{k=\mathrm{K}}$ belongs to the standard K -1 simplex, x being a multinomail parameter and

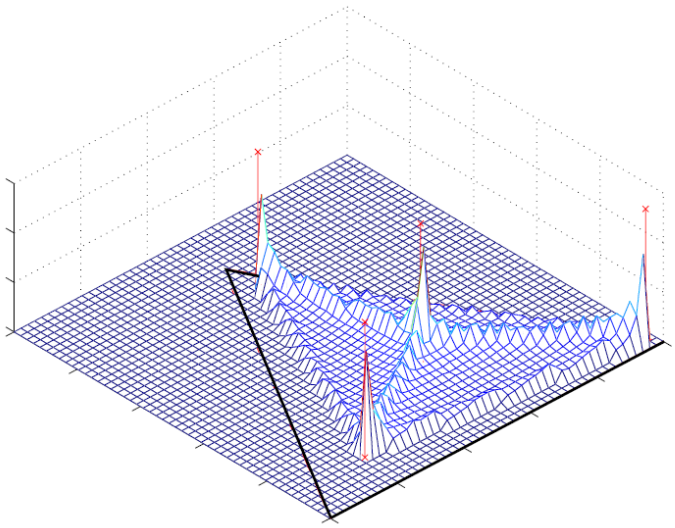

Figure 3.2: Example Dirichlet distribution from [19] that shows an example distribution for a three word and four topic document. The $\times$ marked locations represent the distribution for each of the four topics. This figure helps illustrate the discrete values created when using small Dirichlet hyper-parameters.

K is the number of variables, with

$$\mathrm{B}(\alpha) = \frac{\prod_{i=1}^{\mathrm{K}} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{\mathrm{K}} \alpha_i)} \qquad \alpha = (\alpha_1, \ldots, \alpha_{\mathrm{K}})$$

where α is a defined parameterised vector [34].

The Dirichlet allows us to create a distribution for the latent topics within the document, since the topics themselves are a distribution. As α becomes small, the Dirichlet distribution develops more discrete points representing a sparse topic distribution as mentioned. Figure 3.2 helps to illustrate this by showing a density unigram for a three word and four topic document [19].

3.1.2 Realtime Online Spatiotemporal Topic Modeling

As mentioned in Section 1.3.2, ROST applies LDA to images with the use of visual words instead of text. Visual words are extracted using ORB features and applied to LDA by

having the topic distributions depend on current and previous data. A generative model is used to determine the word and topic observed given the time and location of the observation. A distinguishing factor of the generative model is its use of neighbors to evaluate the topic distribution [10]. These neighbors are observations that are spatially and temporally related to the current observation [10]. To better express the neighbors, the world is broken down into a two-dimensional view as depicted in Figure 3.3. Each image captured is considered an observation. An observation is compared to images from previous times at that current location, and it is compared to images taken at other locations near the current location. This group of neighbors is used to refine topic labels for the current image, which results in a method to learn and compare topics that quickly converges on a topic label [10]. To calculate the probability a word w_i belongs to a topic label k at a location x_i, a Gibbs sampler is used by sampling from the posterior topic distribution:

$$\mathbf{P}(z_i = k | w_i = v, x_i) \propto \frac{n^v_{k,-i} + \beta}{\sum_{v=1}^{V}(n^v_{k,-i} + \beta)} \cdot \frac{n^k_{G_i,-i} + \alpha}{\sum_{k=1}^{K}(n^k_{G_i,-i} + \alpha)}$$

where $n^v_{k,-i}$ counts all words of type v within the topic, excluding the current word v_i, $n^k_{G_i}$ is the number of words within a topic in the neighborhood G_i that also excludes the current word, and α, β are hyper-parameters of the Dirichlet distribution [10]. A realtime Gibbs sampler algorithm is used to add new observations to the topic model [10]. A Gibbs sampler is a framework used to sample from a multivariate distribution by sampling from a conditional distribution rather than a marginalized combination, similar to equation (3.1) [10]. The Gibbs sampler randomly chooses observation times to use for re-sampling topic labels and updating the topic model [10]. This equation is the foundation for the ROST topic modeling.

Having the topics refine over the observations of its neighbors provides a richer understanding for the robot. The robot is more readily equipped to recognize new objects

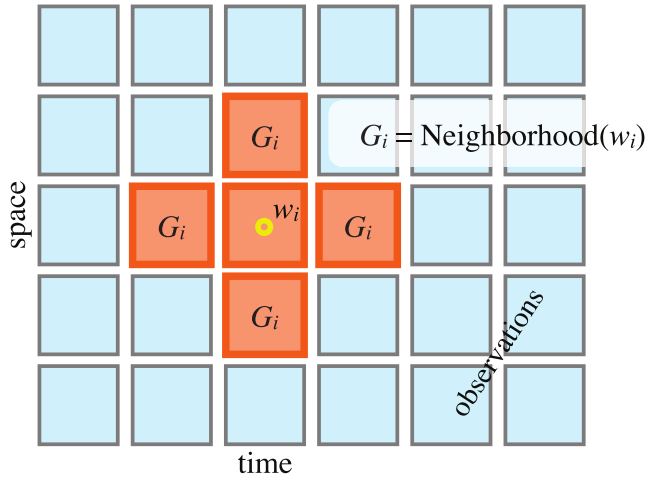

Figure 3.3: Here each cell is an observation. The observation is assumed to have four neighbors, 4 spatial and 2 temporal (2 spatial neighbors not depicted), which allows the model to be used with large datasets, where saving all observations would not be feasible [9]. Each topic is evaluated over all the neighbor cells to provide a more refined topic label for a given word w_i [9].

encountered while traversing through the environment. One metric that the robot can understand a new object is through a perplexity value. In the ROST program it is determined by first segmenting an image into a grid, separate from the spatiotemporal view, normally 20 by 15. Each grid cell is analyzed for a perplexity value, either for visual words or for topics. Specifically in our work, a topic perplexity values is used for guiding the robot's focus. Although, the topic perplexity value is just one weighted component used for decision making in [10]. We focus on only the topic perplexity since our bottom-up focus portion only needs to be able to discern interesting objects within the environment.

Perplexity is calculated for each grid cell of the image using a sample of words observed at g_i with the equation

$$T(g_i) = \exp\left(-\frac{\sum_i^W \log(P(z_i = k|\mathbf{p}))}{W}\right) \tag{3.2}$$

where W is the number of visual words observed, while the perplexity of a particular cell

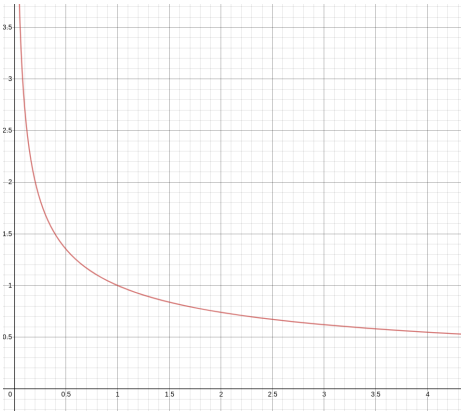

Figure 3.4: A graph to illustrate how Equation (3.2) behaves. The x-axis represents an abstraction of the probability, while the y-axis represents an abstraction of the topic perplexity. The graph helps to illustrate how perplexity can be understood as the inverse of probability.

$T(g_i)$ is calculated by taking the probability that cell is within a particular topic k, based on the previous locations or path $\mathbf{p}$ seen [10]. The temporary topic labels z_i generated are used only for determining perplexity and not added to the topic model [10].

Perplexity can be best understood as the inverse of probability. Equation (3.2) utilizes a log function to represent the data, this data is scaled by taking the average of these values. This log graph representation does not give the desired behavior so the negative exponent is used, which is approximately represented in Figure 3.4. As seen from the figure, as the probability values increase the perplexity approaches zero, and when the probability values approach zero the perplexity increases rapidly.

Perplexity is calculated in realtime with the ROST program. The ROST program, found at [35], provides a user-friendly program called *Sunshine* that preforms the topic modeling analysis for images and video, as in [10]. Sunshine also provides tools to visualize the data. For example, Sunshine was run on an image, Figure 3.7 [36], and the topic perplexity was found as shown in Figure 3.8. The raw perplexity data provided is large and varied. Perplexity values can be high for subjectively uninteresting objects, such as the sand highlighted in white as shown in Figure 3.8. It can be argued that the stingray overall is a more interesting

point of focus. Since there is a disparity between what has the maximum perplexity and what is subjectively more interesting, a methodology for interpreting this data was created. Our perplexity interpretation algorithm uses a grouping and thresholding mechanism to reduce the perplexity data, creating reduced points of focus for a robot.

3.2 Algorithm Methodology - ROST Interpreter

To effectively utilize all the perplexity data generated from ROST ([35]) an algorithm was developed to concentrate the data. The ROST software is able to generate a topic model of the scene while determining the perplexity of the image. The perplexity is based on either the visual words or the topics within the scene. Perplexity is used as a measure of curiosity. Topic perplexity was chosen as the curiosity measure for this project since it more accurately details the high level view of objects in the scene. It is important that the robot recognizes new objects that it observes but remain desensitized to "shiny" objects. To ensure the robot can focus on new objects, we created a method of thresholding to reduce the perplexity values and a grouping method to focus on the largest concentration of high perplexity values.

This program makes some fundamental assumptions that guide its decisions. It is assumed that all the perplexity values are normally distributed across the image. It is also assumed that perplexity values are grouped by row and column, no diagonal cases are considered. Utilizing these assumptions the ROST Interpreter program was created.

3.2.1 Thresholding ROST Perplexity

In order to provide a direct understanding of objects the robot observes, we processed the perplexity data through a thresholding algorithm. This algorithm effectively parsed out low perplexity values and large outlying values. It is important that the robot understands the magnitudes of perplexity, but also the area that the values occur. To adequately represent

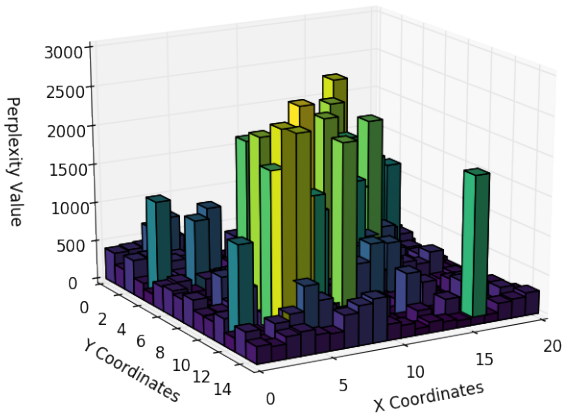

Figure 3.5: This figure expands raw perplexity data generated by the ROST Sunshine program [35]. The data is the same as that seen in Figure 3.8. This view represents the volume used when processing through the threshold. The x and y axes represent the coordinates of the image, with the z axis taken as the perplexity values for each grid coordinate.[2]

this, a volume approach was taken. Taking the volume of the perplexity values allowed the incorporation of the area covered by the object and the perplexity itself. Here the volume, v, is based on the grid area, a_{grid}, the perplexity value covers multiplied with the perplexity value, λ_{topic} [2].

$$v = a_{\text{grid}} * \lambda_{\text{topic}}$$

A volume view of the perplexity data from Sunshine can be seen in Figure 3.5 [2].

The volume of the perplexity data was then filtered through the defined threshold. It was assumed that the topic perplexities are normally distributed over any image. Using this assumption, each image threshold values were set as

$$(\sigma_{x_{\text{ppx}}} + \overline{x}_{\text{ppx}}) < \mathbf{x}_{\text{ppx}} < (2 * \sigma_{x_{\text{ppx}}} + \overline{x}_{ppx})$$

where $\mathbf{x}_{\text{ppx}}$ represents the accepted perplexity values, $\sigma_{x_{\text{ppx}}}$ represents the standard deviation

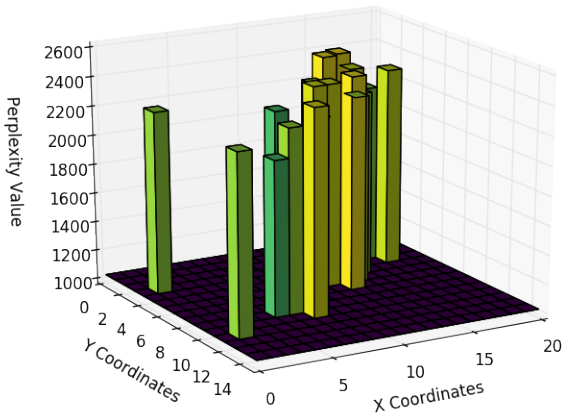

Figure 3.6: This volume view takes the data from Figure 3.5 and applies the threshold equations to it. The bars now represent the locations of points in the image with sufficient perplexity. The points with high perplexity are removed as well as low perplexities [2].

of all the perplexity values, and $\overline{x}_{\text{ppx}}$ is the mean of the perplexity values [2]. Extracting this portion of the perplexity data accurately represents interesting objects by removing strong outliers and capturing significantly perplexing areas. Figure 3.6 demonstrates the volume data after applying the threshold to the data shown in Figure 3.6 [2]. The topic perplexity reduces as the robot makes observations over time. So as the robot traverses through the environment, the threshold area to be triggered when an unknown object enters the scene will be more clear than just the perplexity data given by the Sunshine program [2].

In order to ensure the perplexity mean, $\overline{x}_{\text{ppx}}$, was representative of the environment, a running average is maintained. Initially the mean is calculated using an initial 30 seconds of the first image observation [2]. This allows for the current topics to be adequately learned, which stabilizes the perplexity values. As new perplexity data is received the mean, $\overline{x}_t$, is updated at each time-step by $\overline{x}_t^* = \frac{\overline{x}_{t-1}+\overline{x}_t}{2}$ [2].

The applied threshold provides a reduced data set that allows for better online decision making. Removing the low perplexity topics provides a better point for the robot to focus on

and navigate toward. Our interpretation of the perplexity data makes several assumptions that would need to be analyzed to ensure the threshold provides the best point of focus for the robot [2].

3.2.2 Grouping ROST Perplexity

In addition to the thresholding, a grouping algorithm was created so the area covered by an object is also considered. Grouping together the perplexity values creates distinct locations of high perplexity objects, where the group with the most area covered is considered the interesting location. In essence, the program takes the topic perplexity data and returns the coordinates of the most perplexing object in the current image. The threshold perplexity values are grouped together based on the spatial relation between them. As mentioned, values are grouped together if they are connected horizontally or vertically. These groups are used to determine a point of focus for the bottom-up case.

As the perplexity values are processed through the threshold, the position of each point meeting the criteria is saved. Algorithm 2 shows the logic flow for grouping the values. This algorithm takes a sliding window approach. The array of perplexity values, is scanned sequentially to extract values and group them.

Algorithm 2: Perplexity Grouping

```
initialization: c (array of (x coordinate, y coordinate, perplexity value));
while new perplexity data do
    Calculate Running Average
    if Time > 30 sec then
        for Length of c do
            if Perplexity within Threshold then
                Add c_i to group list g_i
                Look at neighbors ((x − 1, y), (x + 1, y), (x, y − 1), (x, y + 1))
                Store Looked at positions
                if Neighbor not looked at and meets threshold then
                    Add c_i, of neighbor, to group list g_i
                end
                #Check if current value is within the same group
                if abs((c_{i,x} − c_{i,x−1}) + abs((c_{i,y} − c_{i,y−1}) > 1 then
                    g_i = g_{i+1}
                end
            end
        end
    end
end
```

Once all the data is processed, the groups are searched for a maximum value using built in Python tools. The location of the group with the largest perplexity value is then published for use in other programs.

3.3 Testing

To test the viability of the thresholding and grouping algorithms, the processes were run on underwater images. Figure 3.7 is an example image that was chosen for testing. The image contained one user-defined interesting object, the stingray, with an uninteresting background. The image was processed for approximately ten seconds with the ROST Sunshine program, resulting in Figure 3.8. As the figure shows, the Sunshine program does capture the stingray as interesting, represented by the red points, but also includes some subjectively uninteresting objects like the yellow spots, or parts of the sand. This perplexity data was then processed through the ROST Interpreter algorithm. The ROST Interpreter provided a more adequate capture of the stingray. This can be observed by comparing Figure 3.8 with Figure 3.9. As seen in Figure 3.9, the majority of remaining perplexity values lie on the stingray, which is subjectively more interesting than the sandy area denoted by the maximum value in white, as seen in Figure 3.8. Additionally, Figures 3.5 and 3.6 contain the same perplexity data from the image and are provided to show the volume view used when processing [2].

Furthermore, Figure 3.10 show the processing of different images yielding similar results. These figures help to illustrate how applying a threshold mechanic on the perplexity data can extract more relevant objects for the robot's focus. Although, the ROST Interpreter algorithm was unable to capture objects when they were relatively small. This resulted in the object not being sufficiently captured, as shown by the top of Figure 3.11. One way to mitigate this issue was to modify the thresholding parameters. For instance, the fish in Figure 3.11 was captured after thresholding by increasing the accepted top end perplexity values to $(2.5 * \sigma_{x_{\text{ppx}}} + \overline{x}_{ppx})$, resulting in the bottom of Figure 3.11. Because of this issue, the accepted threshold range would likely need to be optimized for the specific application it is applied to.

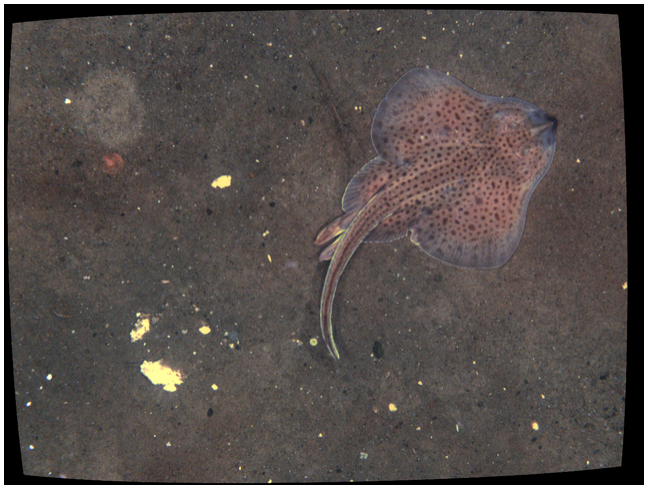

Figure 3.7: This test image was obtained from [36] and used for processing with ROST Sunshine, and the ROST Interpreter [2].

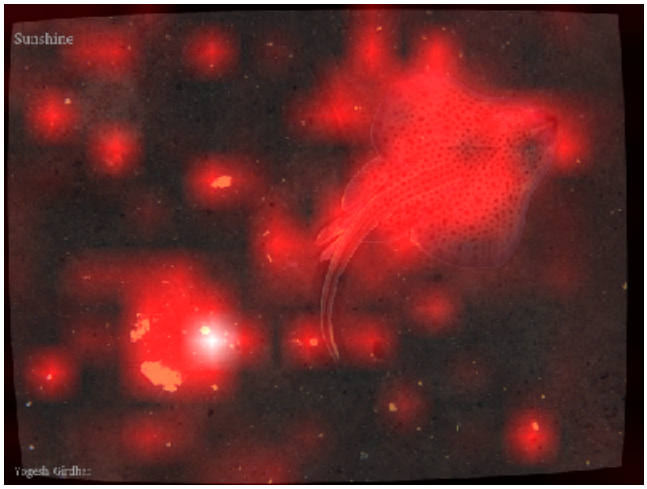

Figure 3.8: The Sunshine program from ROST was run on an image of a stingray from [36] [35]. Topic perplexity was calculated for each part of the image as the program analyzed it for several seconds, as shown in red. The white dot represents the point with the highest topic perplexity [2].

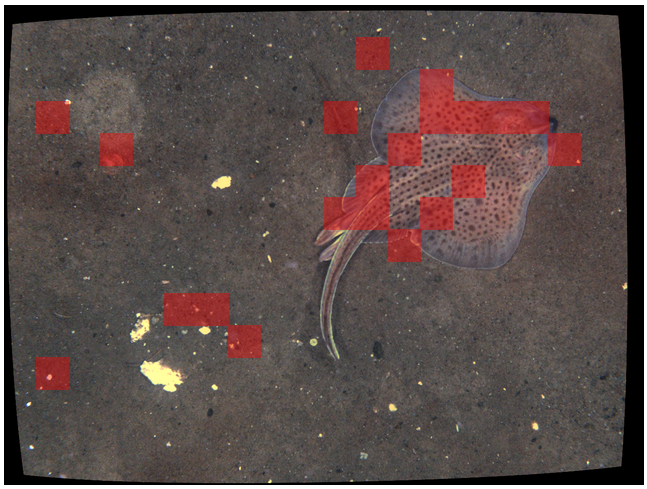

Figure 3.9: The raw perplexity data from Figure 3.8 was run through the threshold algorithm developed to reduce the data, leaving a better representation of interesting objects. The perplexity data visualized represents the locations of perplexity that meet the threshold criteria. By removing large outliers and low value perplexities the robot now has a sparse representation of objects in the image on which the robot will focus [2].

3.4 Conclusion

The proposed ROST Interpreter program expands upon the existing ROST functionality to provide a minimal representation of curious objects within a scene. Applying a threshold preserves the perplexity data needed to adequately determine curious objects. Grouping these values together incorporates the area covered by an object into navigation decisions. The program provides a refined point of focus for the bottom-up element in the hybrid focus model.

The next chapter will provide insight on how the hybrid focus model was created and tested. The implemented methodology for the top-down and bottom-up focus modes can then be combined to generate the robot focus.

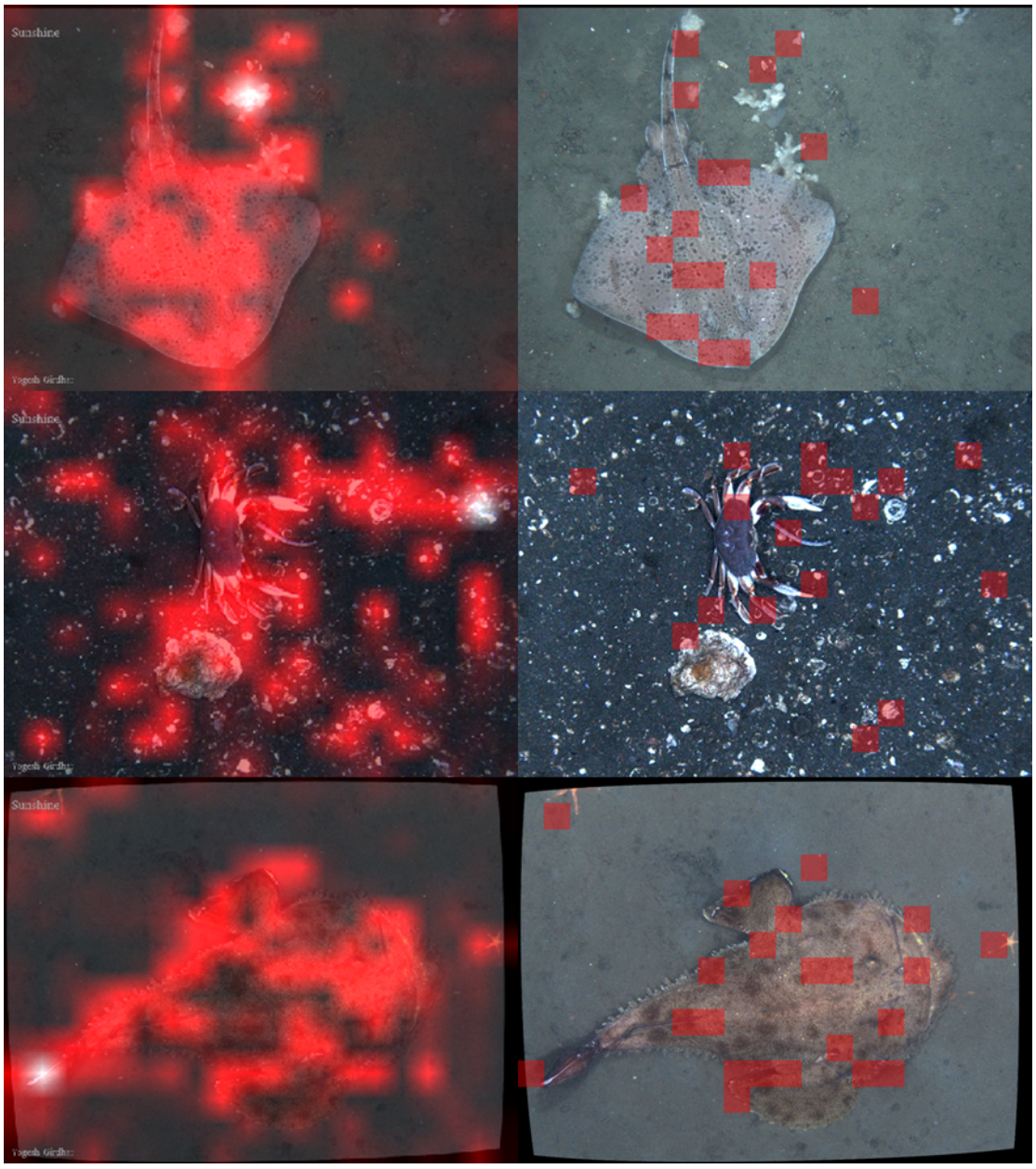

Figure 3.10: The images on the left hand side is the perplexity data as given by the ROST Sunshine program, 3.8, while the right hand side is the perplexity data after being processed through the ROST Interpreter. Here we demonstrate that the ROST Interpreter reduces the perplexity data, while the interesting objects remain identified. In order to illustrate that simply using the maximum perplexity is not sound, each image has one interesting object with a dull background. Reducing the perplexity data better captures the interesting objects. All images obtained from [36].

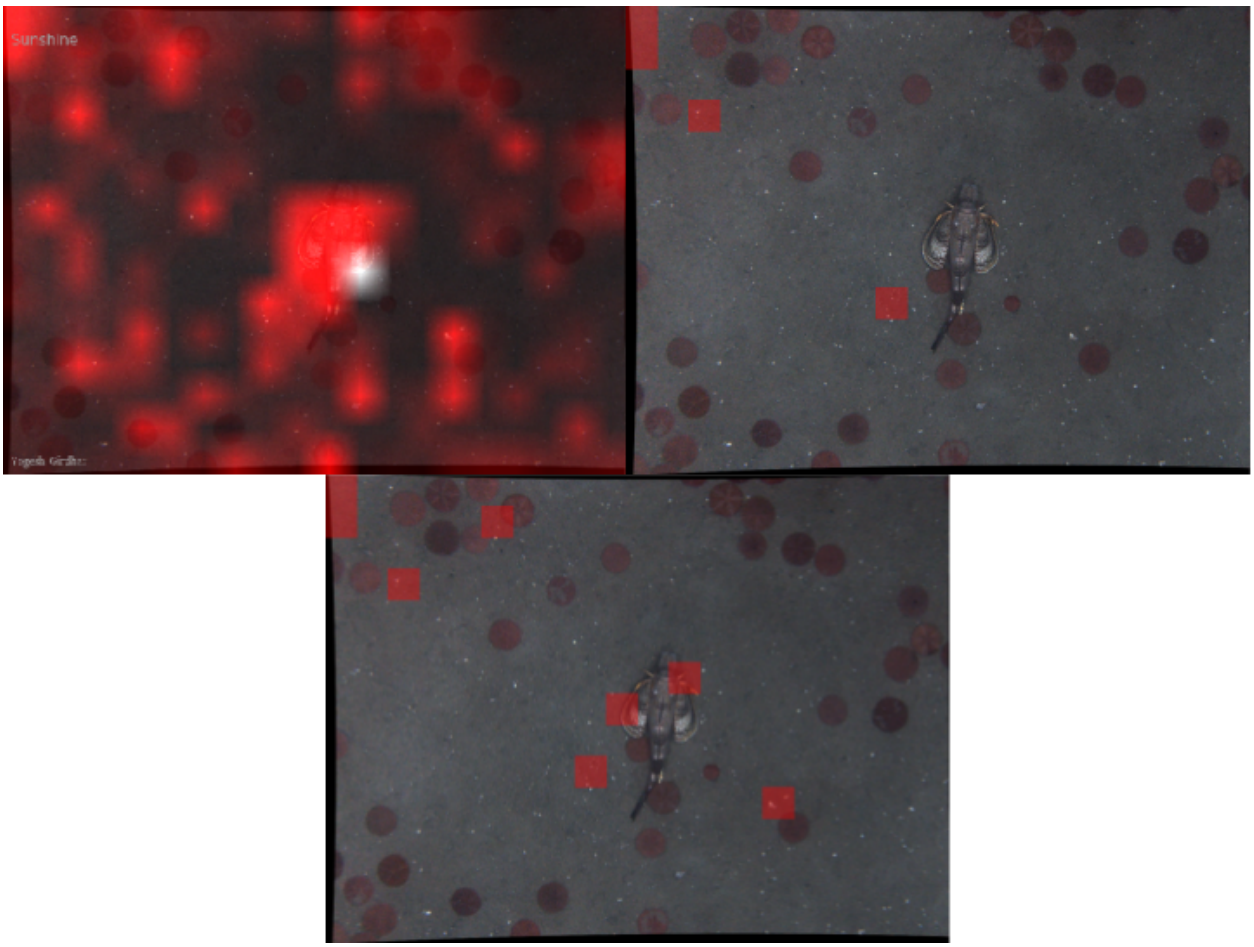

Figure 3.11: Due to the small size of the object of interest the object was inadequately captured after thresholding the perplexity values. The bottom image shows the perplexity values obtained after modifying the threshold range. It would be important to optimize the thresholding range given the specific application, but the default values do apply in the general case. The image was obtained from [36].

CHAPTER 4
HYBRID FOCUS MODEL

4.1 Introduction

To better facilitate ocean exploration our goal is to create a hybrid focus model that leverages curiosity from a bottom-up approach and prior knowledge to make robust decisions for exploration. We aimed to emulate a human like focus, as in [11], to look at certain points in order to maximize information gain using both salient points and prior knowledge. Using a hybrid model will allow the robot to smartly assess the environment and explore new findings.

The hybrid focus model, as shown in Figure 1.2, relies on image sensing to extract information. This information is used for both the bottom-up and top-down focus processes as described in Chapters 2 and 3. Figure 1.2 shows the total concept of the hybrid focus model, while the following describes the lighter colored regions. This algorithm was developed to test the concept of a hybrid focus model using underwater video and an indoor robotic platform.

4.2 Algorithm Methodology - Hybrid Focus Model

There are many approaches in prioritizing decisions. This version of the hybrid focus model uses curiosity (i.e., perplexity values) as the main driver of focus. The attention of the robot is directed first by the perplexity of an object, then if the object of attention is known by the object detector, the focus shifts to the next perplexing object. Here we have simplified the combination of the two focus models to test a proof of concept.

4.2.1 Hybrid Combination

Each image is analyzed with the ROST Interpreter algorithm and YOLO algorithm. Both algorithms make use of bounding boxes to represent locations in the image. These boxes are compared to make use of both sources of information for determining a focus region. The focus region is updated each second to reflect a consistent decision making routine, since running decision making process continuously at a fixed frequency produces better results than at the maximum frequency possible [37].

The perplexity data is obtained independently by running the ROST Sunshine program separately. Normally these processes would run in parallel but interfacing directly with the Sunshine program was not possible with Python. As such, perplexity information is generated for each image and imported into the algorithm through a text file. Algorithm 3 shows the simple logic flow used to process each image. The algorithm determines a focus region at the maximum perplexity group. A combination of both focus models is achieved with the use of bounding boxes. A focus box, of a default size (96×96 pixels) is generated around the group of the highest perplexity as determined by the ROST Interpreter. This focus box is compared to any bounding box generated by YOLO. If the boxes overlap a new focus box is generated around the next highest perplexity group, then overlap is reassessed. Overlap between the boxes is determined by comparing the coordinates of the box corners. Given two boxes A and B the following rules dictate if the boxes intersect.

Table 4.1: Intersection Logic Criteria for Boxes

Logic Criteria
$A_{x_L} < B_{x_R}$
$A_{x_R} > B_{x_L}$
$A_{y_L} < B_{y_R}$
$A_{y_R} > B_{y_L}$

In Table 4.1 (x_L, y_L) refers to the coordinates at the top left corner of a box, and (x_R, y_R) refers to the coordinates at the bottom right corner of a box. An intersection occurs if all of these criteria are true. Alternatively, if any one criterion is false the boxes do not intersect. This is a common use of logic to determine rectangle intersections as described in [38].

Algorithm 3: Hybrid Focus

```
1 while new image do
2     Process perplexity values with ROST Interpreter
3     Process image with trained YOLO model
4     Focus box drawn at maximum perplexity group
5     while Focus box and YOLO bounding box intersect do
6         Draw focus box at next highest perplexity group
7     end
8     Output overlays onto image
9 end
```

4.3 Testing

The hybrid focus algorithm was run on an underwater video from [39]. This video was chosen for having a large diversity of aquatic animals with a focus of one or two animals clear in the foreground. The video was also used to generate images for training with YOLO, where the stingray served as the knowledge base for the top-down focus.

The algorithm was also used during exploration with a TurtleBot3 Burger in a laboratory setting. The TurtleBot3 was used in an area with sparse objects to map a known and unknown object.

4.3.1 YOLO Training

In order for the YOLO algorithm to recognize the stingray within the video, a CNN model had to be trained. At first a collection of approximately 60 stingray images were gathered for training. Each image was hand labeled with a bounding box defining the area with the stingray. To generate the bounding box data files a tool from [40] was used. Ten percent of the data was held out for testing, with the rest used for training.

The Fast YOLO CNN was trained with the training data. This CNN is similar to the normal YOLO CNN but has fewer convolutional layers [14]. The fast YOLO was chosen based on available hardware and to reduce the training time.

The network was trained following a guide from [41]. The CNN was trained over 2000 iterations of the training data, with the images randomly used in batch sizes of 64. Once the training was complete, the model was used on images from the testing video. YOLO was unable to consistently recognize the stingray within the video. It was determined that the training data was too small and varied for the model to properly learn. Since creating the training data was a cumbersome process and not the basis of this work, images from the video were directly used. A set of 130 hand labeled images were generated from the video for training. The fast YOLO model was trained using the same parameters. These images caused the YOLO model to overfit to the data, resulting in the stingray being reliably recognized. Using the same training and testing data is certainly not a normal practice for machine learning, but a generalized model was not needed for this test.

4.3.2 Video Simulation

The testing video was converted to a series of images using the ffmpeg program. The Linux command `ffmpeg -i SimulationVideo.mp4 thumb%04d.jpg -hide_banner` was used to convert each frame to an image.

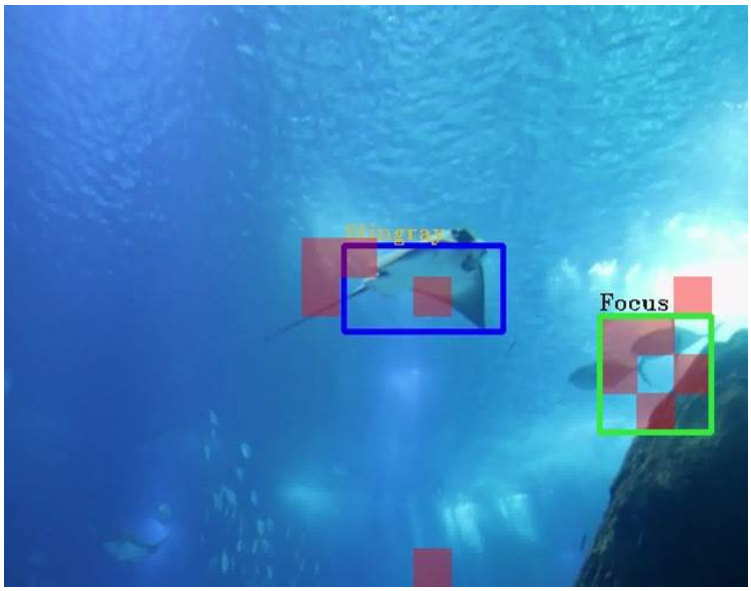

Figure 4.1: Result of running the hybrid focus model on an underwater video. Here the bottom-up focus is represented by the red squares, with the top-down focus represented with the blue bounding box. The green focus box shows the point of attention of the robot, which observes the group of fish instead of the already recognized stingray.

As described, the ROST Sunshine program was run separately to generate perplexity data. In order to establish a base average perplexity, Sunshine was run on the first frame of the video for 30 seconds. Using this average perplexity, the hybrid focus algorithm processed each frame of the video. The running average of perplexity was maintained as each image was processed, as described in Section 2.2. The video is processed through the hybrid combination algorithm frame by frame. Once a focus region is determined, the bounding boxes and perplexity data are overlaid onto each frame and output using OpenCV tools.

Figures 4.1 and 4.2 are selected images from the generated hybrid focus algorithm. The focus region, displayed in green, observes both regions with high perplexity, red squares, and does not intersect with already known objects, shown by the blue box. These figures demonstrate the proof of concept of using the focus box to direct a robot's navigation.

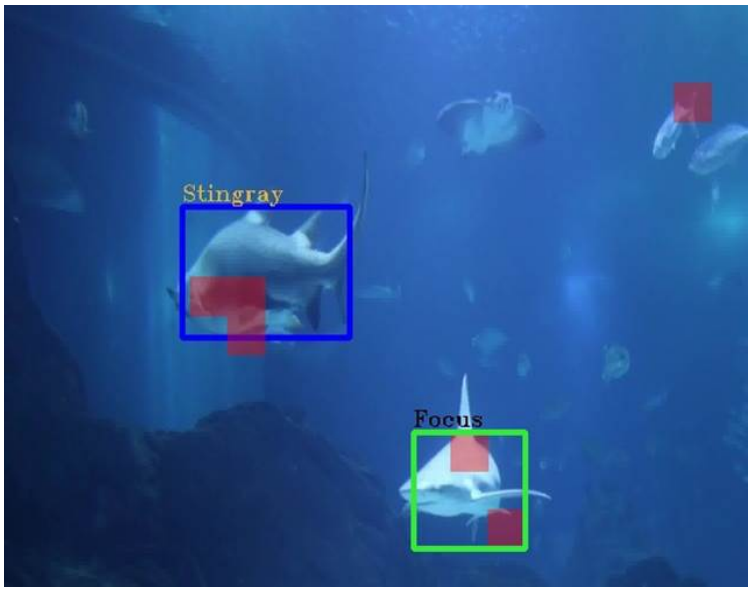

Figure 4.2: This result is similar to that of Figure 4.1, but as shown the shark is incorrectly recognized as a stingray. This is due to a lack of training data for the YOLO model. Regardless, the focus of the robot is still valid since it observes the other shark instead of a recognized area, the stingray box.

4.3.3 Semantic Mapping

The hybrid focus model was tested on the TurtleBot3 by continuing a similar approach as in Section 2.3.2. The goal of the test was to correctly mark both the known object and novel object within the occupancy grid map. To meet this goal some modifications to the TurtleBot3 were made. The challenges of slow image processing met in Section 2.3.2 provided motivation to upgrade the processing power of the TurtleBot3. The TurtleBot3 CPU was upgraded to a Jeston Nano.

TurtleBot3 Setup

The Jetson Nano maintains a similar footprint to the Raspberry Pi 3, but has a built in GPU. To process the images more rapidly, this GPU was utilized for running the darknet ROS package. To do so, the ROS software had to be upgraded to ROS Melodic. This different version of ROS was not fully compatible with all the previous packages used, mainly the

Raspberry Pi camera. The camera was replaced with a Logitech Webcam C270. All other programs and hardware were able to remain the same as in Section 2.3.2.

Despite the increased processing power, the Jetson Nano was unable to run both the darknet ROS and ROST Sunshine programs needed. ROST Sunshine was instead run on a laptop with an i7 processor. This separation required the images from the camera on-board the TurtleBot3 be saved separately for access by the ROST Sunshine program. A ROS node was created for this purpose, along with the nodes needed for pre-planned navigation, and the ROST Interpreter. Figure 4.3 gives a high level view of the processes running in ROS for this test.

Environment Setup

The environment was setup to contain two objects: a bottle, serving as a top-down focus object, and a yellow bucket, serving as the bottom-up focus object. The area was constructed with a cardboard wall in an approximately 2×2 meter square. Initially the testing area was mapped with the TurtleBot3 without any objects present to allow for the pre-planned navigation node to process. This pre-planned route was made to ensure the robot would adequately traverse and view the two objects within the environment. The same SLAM, and object detection programs were used, as discussed in Section 2.3.2, but in this test the ROST Sunshine program was run in parallel to provide topic perplexity information. Perplexity was processed through the ROST Interpreter as described in Section 3.2. The locations of interpreted perplexities and detected objects of interest were both sent to the mapping algorithm, as discussed in Section 2.2. [2]

Performance

Initially, the robot remained stationary for 30 seconds, allowing a baseline topic perplexity to be obtained, while maintaining a running average every frame to provide a representative

topic perplexity of the environment. As the robot traveled across the environment, it was able to identify each object when it was encountered visually, being represented by a marking within the occupancy grid map [2]. Figure 4.4 and Figure 4.5 show the known object, a bottle, and the new novel object being marked in yellow and red, respectively [2]. The were some issues encountered with ROS while preforming the test, so although the top-down and bottom-up processes were run together, the output had to be shown separately. It is also important to note that errors and outliers still existed when mapping and recognizing the objects. For example, as seen in Figure 4.5 the ROST Interpreter was incorrectly triggered and mapped to incorrect locations before the bucket was observed [2].

Despite the issues, the proof of concept is shown. The recognition of the novel object could be used as a trigger for making navigation changes based on what it has perceived. For example, once the yellow bucket was encountered it could deviate from the pre-planned path to then circle the object until the perplexity is reduced below the threshold, spending less time observing the already known bottle [2].

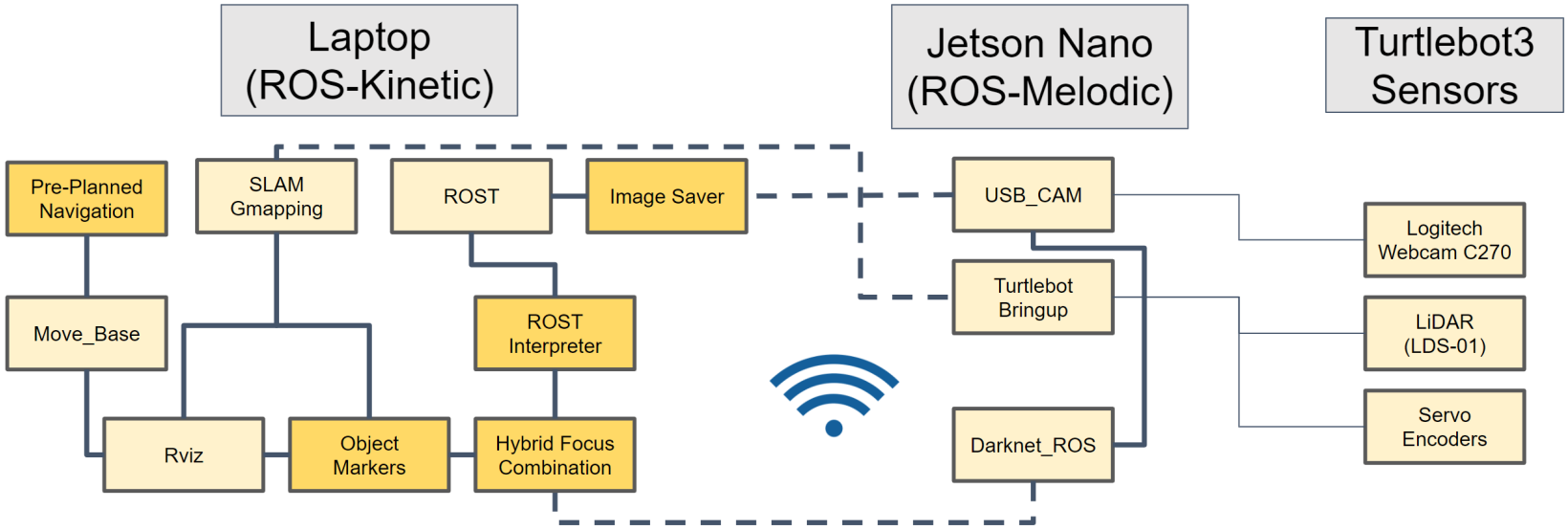

Figure 4.3: Each box represents a separate ROS node used in the testing of the hybrid model. The light yellow boxes represent the nodes and programs created for this work, while the tan boxes are pre-built packages in ROS. The dashed lines represent communication over a wireless network, while the solid lines represent directly connected communication. More information about the pre-built packages can be found at [30] and [27], or the created nodes at `https://github.com/song-ranlab/curisoity_exploration_and_hybrid_focus`.

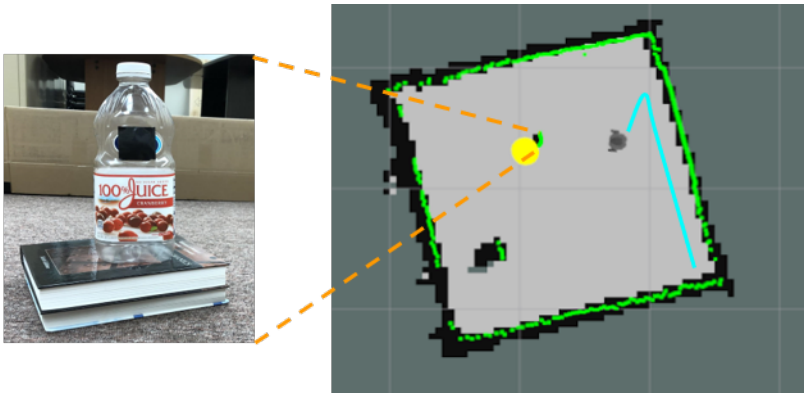

Figure 4.4: Results of the hybrid model testing on the top-down focus. The robot was able to correctly identify the trained on object, a bottle, and correlate it onto the map. This region would not be investigated by the robot since it already knows the object.[2]

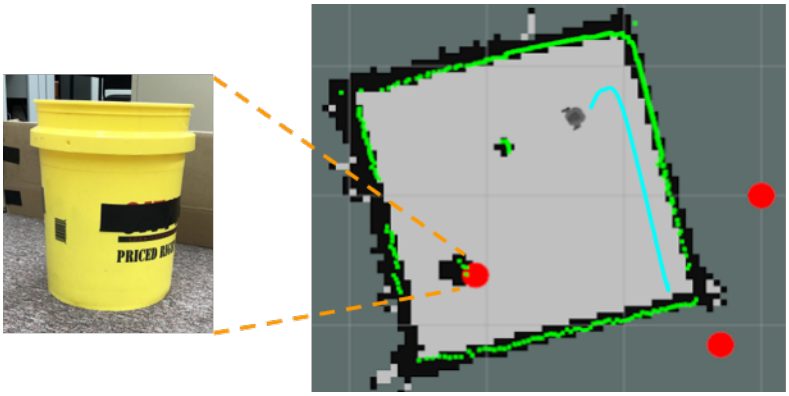

Figure 4.5: Results of the hybrid model testing for the bottom-up focus. As soon as the robot discerned the yellow bucket, the perplexity values rose enough to trigger the focus and label the bucket with the red marker. Some outlier observations were marked while traveling as shown by the red markers not within the explored area.[2]

4.4 Conclusion

The hybrid focus algorithm demonstrates a novel way to combine both the top-down and bottom-up focus strategies. Video simulation and semantic mapping testing show how the model is a viable method to use for decision making. Observations are enhanced above a pre-planned trajectory model since the hybrid focus model facilitates observing interesting objects and avoid uninteresting locations.

In the scenario demonstrated, the robot would navigate to observe objects in the focus region, while limiting time spent observing uninteresting or known regions. Alternatively, the robot could use the known object location as an observation goal until the perplexity at that location decreases and a new navigation goal determined. The hybrid focus model could be employed in a variety of ways depending on the exploration goal of the application. Furthermore, the semantic labels generated from both focus methods provide a more enriched map, better facilitating robot and human interaction.

CHAPTER 5
CONCLUSION

5.1 Summary

We have proposed a hybrid focus methodology approach for exploration decision making in an underwater environment. This hybrid focus model relies on two sources of information: top-down focus, and bottom-up focus. The top-down focus is facilitated by a realtime object classification algorithm, here we use the YOLO program. The object classification algorithm utilizes previous knowledge to locate objects within an image. This mimics the top-down focus in humans since it uses a knowledge base to understand what is being observed. The bottom-up focus uses low level features within an image to discover salient points. In our implementation we take a higher level look at these low level features with the use of topic modeling. Topic modeling provides a semantic framework to classify different parts of an image. This higher level view allows us to leverage a curiosity style function to direct the robot. The curiosity is measured by a topic perplexity value formulated by the ROST methodology and program. We then combine these information streams for use as a point of focus. This focus point can be used to affect the robot's behavior to explore and learn new objects. Our methodology was validated by utilizing underwater images, video, and implementation on a land based robot in a laboratory setting.

In addition, the model can be used to semantically label a map to provide a more clear understanding of what the robot has seen. Using a combination of computer vision and a range sensor, the type of object observed can be correlated to be mapped at the observed locations. In our testing, we validated the ability to map objects of interest and curious objects. Markers for these objects were overlaid onto an occupancy grid map generated by an indoor robot preforming SLAM. This semantic map provides enhanced understanding

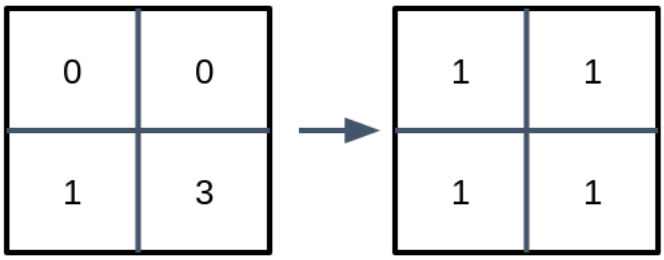

Figure 5.1: The large box represents the window that would slide over the perplexity data. Each grid inside the box represents the perplexity data at a observed location within the image. These values would then be averaged and distributed within the window, as shown by the box on the right. This technique would generate a higher quality representation of the perplexity data.

of the robot's observations during exploration, allowing patterns or locations to be easily recognized.

5.2 Future Work

This work can be further improved upon by investigating several concepts for better implementation on an underwater platform. The following sections provide concepts on how to improve this work.

5.2.1 Perplexity Averaging

To further improve the ROST Interpreter algorithm it would be beneficial to reprocess the perplexity data utilizing a sliding window. This window would cover the entire image to better distribute the perplexity data. By averaging the perplexity values within the window would allow outliers to be reduced without discarding any information. In this way, the perplexity data is averaged and smoothed across the entire image.

Figure 5.1 demonstrates this concept. Each box represents a perplexity value for a given portion of the image. The average of these perplexity values would be calculated and dis-

tributed evenly throughout the window, while maintaining the same average. Furthermore, the perplexity values can also be normalized by using a log function. For example, $\frac{1}{log(ppx_x)}$ can be used to normalize the data, where ppx_x is the perplexity value at grid location x. Applying these concepts would allow the perplexity data to be represented in a way that the information is compressed without removing any data.

5.2.2 Temporal Averaging for Focus Region

Temporal averaging for determining the focus region can be implemented. As detailed in the hybrid focus video testing, section 4.3.2, the focus region was updated at one second intervals. The focus region should account for regions covered in the previous time steps instead of directing focus on just the current observation. Taking the average region where the focus was directed over a given time period could improve the determination of the focus region.

5.2.3 Computational Complexity

The computational complexity of the algorithm can be reduced through more efficient algorithm development. The use of both an object detector and the ROST program require large processing power. With the addition of the ROST Interpreter, the complexity is further compounded. Making the auxiliary programs as efficient as possible will lessen the computational burden needed to run all the processes simultaneously.

Another potential solution would be to use a lens filter on the camera to reduce the brightness of any reflective objects. Using a filter could remove outlying objects without the need for a software solution. The use of physical components to preform the same functions as portions of the software would reduce the computational burden.

5.2.4 Control Scheme

A control system or scheme for the autonomous robot can be built upon the hybrid focus model. Since the robot will be in motion during observations, a method to predict where the focus location will be can help assist in the robot's control. Through predicting the focus region the robot can anticipate maneuvers rather than react to the focus region. This concept is illustrated in Figure 1.3.

5.2.5 Topic Labels

Normally, the topic labels generated are sequential numbers. These labels could be enhanced with semantic labels by comparing the perplexity region and object classification region. The topic within the region covered by the object classifier could be labeled as the detected object. Adding semantic labels to the topics can help facilitate human understanding of the topics observed during exploration.

APPENDIX A
OCEANS SEATTLE POSTER

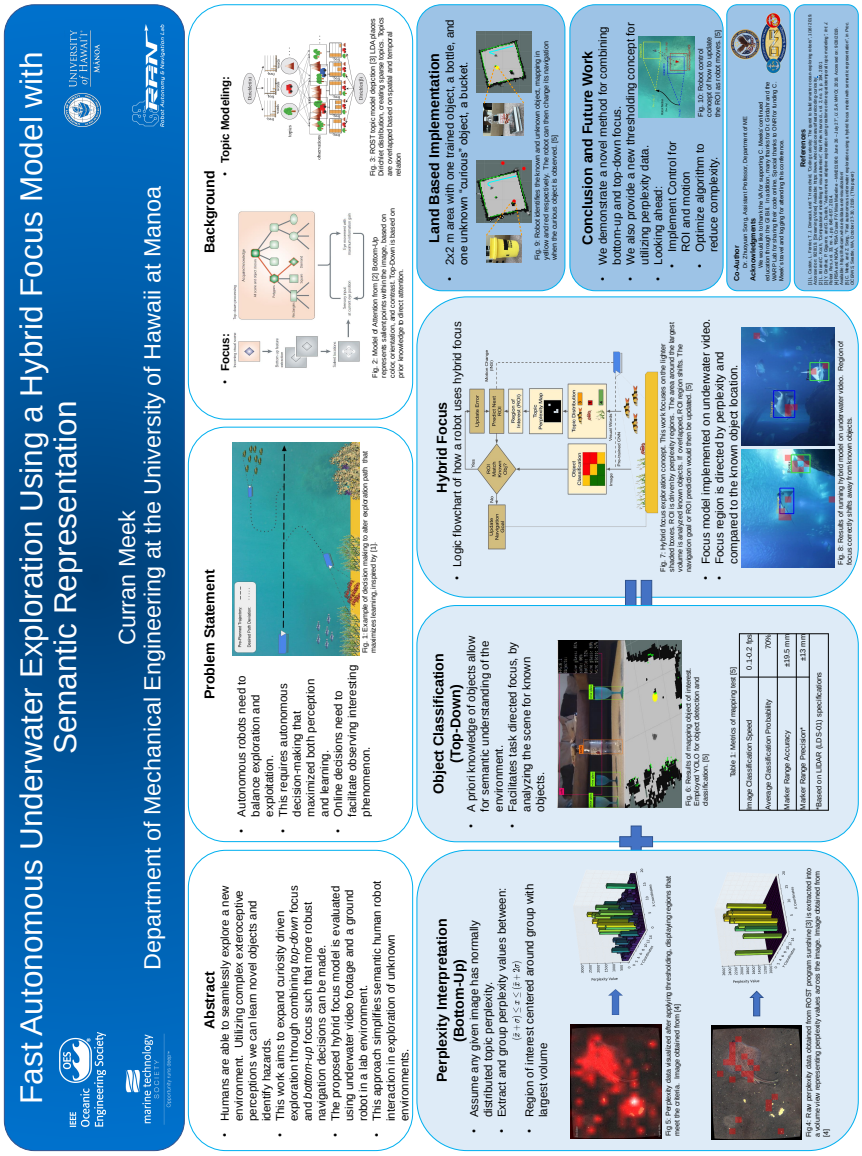

Figure A.1: This poster describes an overview of all the work within this thesis. It was submitted in the Student poster competition in Seattle Washington 2019.

APPENDIX B
CODE REFERENCE

All the code generated in the development of this work can be found at:

`https://github.com/song-ranlab/curisoity-exploration-and-hybrid-focus`

Included readme file details how the programs were used and links to resources used.

BIBLIOGRAPHY

[1] Laura Castanon, Lochie Ferrier, Timothy James Dimacali, and Tristan Honscheid. Coding curiosity. `https://www.whoi.edu/oceanus/feature/coding-curiosity/`, January 2019. Accessed 5-24-2019.

[2] Curran Meek and Zhuoyaun Song. Fast autonomous underwater exploration using a hybrid focus model with semantic mapping. In *OCEANS 2019 Seattle*, 2019.

[3] Cameron Fabbri, Md Jahidul Islam, and Junaed Sattar. Enhancing underwater imagery using generative adversarial networks. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 7159–7165, 2018.

[4] Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robot. Auton. Syst.*, 66(C):86–103, 2015.

[5] Lawrence W Stark, Claudio M Privitera, Huiyang Yang, Michela Azzariti, Yeuk Fai Ho, Theodore T Blackmon, and Dimitri A Chernyak. Representation of human vision in the brain: How does human perception recognize images? *J. Electron. Imaging*, 10(1):123–152, 2001.

[6] Wenguan Wang and Jianbing Shen. Deep visual attention prediction. *CoRR*, abs/1705.02544, 2017.

[7] Jacqueline Gottlieb, Pierre-Yves Oudeyer, Manuel Lopes, and Adrien Baranes. Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends in Cognitive Sciences*, 17(11):585–593, 2013.

[8] Oleksii Zhelo, Jingwei Zhang, Lei Tai, Ming Liu, and Wolfram Burgard. Curiosity-driven exploration for mapless navigation with deep reinforcement learning. *arXiv preprint arXiv:1804.00456*, 2018.

[9] Yogesh Girdhar, Philippe Giguere, and Gregory Dudek. Autonomous adaptive exploration using realtime online spatiotemporal topic modeling. *The International Journal of Robotics Research*, 33(4):645–657, 2014.

[10] Yogesh Girdhar and Gregory Dudek. Modeling curiosity in a mobile robot for long-term autonomous exploration and monitoring. *Autonomous Robots*, 40(7):1267–1278, 2016.

[11] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3):194, 2001.

[12] A beginner's guide to attention mechanisms and memory networks. `https://skymind.ai/wiki/attention-mechanism-memory-network`. Accessed 5-4-2019.

[13] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[14] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[15] Wenguan Wang and Jianbing Shen. Deep visual attention prediction. *IEEE Transactions on Image Processing*, 27(5):2368–2378, 2017.

[16] Shashank Mujumdar, Nithya Rajamani, L Venkata Subramaniam, and Dror Porat. Efficient multi-stage image classification for mobile sensing in urban environments. In *2013 IEEE international symposium on multimedia*, pages 237–240. IEEE, 2013.

[17] Tao Kong, Anbang Yao, Yurong Chen, and Fuchun Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 845–853, 2016.

[18] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[19] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[20] Xiaogang Wang and Eric Grimson. Spatial latent dirichlet allocation. In *Advances in neural information processing systems*, pages 1577–1584, 2008.

[21] Zhenxing Niu, Gang Hua, Le Wang, and Xinbo Gao. Knowledge-based topic model for unsupervised object discovery and localization. *IEEE Transactions on Image Processing*, 27(1):50–63, 2017.

[22] Zhenxing Niu, Gang Hua, Le Wang, and Xinbo Gao. Knowledge-based topic model for unsupervised object discovery and localization. *IEEE Transactions on Image Processing*, 27(1):50–63, 2018.

[23] G Lowe. Sift-the scale invariant feature transform. *Int. J*, 2:91–110, 2004.

[24] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[25] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, volume 11, page 2. Citeseer, 2011.

[26] Cipriano Galindo, Alessandro Saffiotti, Silvia Coradeschi, Pär Buschka, Juan-Antonio Fernandez-Madrigal, and Javier González. Multi-hierarchical semantic maps for mobile robotics. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2278–2283. IEEE, 2005.

[27] Amanda Dattalo. Introduction. `http://wiki.ros.org/ROS/Introduction`, 2018. Accessed 10-14-2019.

[28] Marko Bjelonic. YOLO ROS: Real-time object detection for ROS. `https://github.com/leggedrobotics/darknet_ros`, 2016–2018. Accessed 10-1-2019.

[29] Markers: Sending basic shapes (c++). `http://wiki.ros.org/rviz/Tutorials/Marker\%3A\%20Basic\%20Shapes`, 2018. Accessed: 10-8-2019.

[30] Robotis emanual turtlebot3. `http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/#turtlebot3`. Accessed: 10-9-2019.

[31] Allen Gut. *Probability: A Graduate Course (Second ed.)*. Springer, 2013.

[32] Eliezer S. Yudkowsky. An intuitive explanation of bayes' theorem. http://yudkowsky.net/rational/bayes/, 2006. Accessed 11-6-19.

[33] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, 42(1-2):177–196, 2001.

[34] Norman Johnson Samuel Kotz, N. Balakrishnan. *Continous Multivariate Distributions*. Wiley, 2000.

[35] Yogesh Girdhar. Realtime online spatiotemporal topic modeling. https://gitlab.com/warplab/rost-cli, 2019. Accessed 9-7-2019.

[36] RSA and NOAA. RSA Cruise F/V Miss Madeline – MM201906: June 26 – July 27. Woods Hole Oceanographic Institution, https://habcam.whoi.edu/data-and-visualization/, 2018. Accessed on: 9/28/2019.

[37] Francesco Amigoni, Alberto Quattrini Li, and Dirk Holz. Evaluating the impact of perception and decision timing on autonomous robotic exploration. In *2013 European Conference on Mobile Robots*, pages 68–73. IEEE, 2013.

[38] Aman Gupta. Find if two rectangles overlap. https://www.geeksforgeeks.org/find-two-rectangles-overlap/, 2019. Accessed 10-12-2019.

[39] Magda Ehlers. Underwater footage. https://www.pexels.com/video/underwater-footage-2556894/, 2018. Accessed 10-12-2019.

[40] Manivannan Murugavel. Yolo-annotation-tool-new. https://github.com/ManivannanMurugavel/Yolo-Annotation-Tool-New-, 2019. Accessed 10-12-2019.

[41] Nils Tijtgat. How to train yolov2 to detect custom objects. https://timebutt.github.io/static/how-to-train-yolov2-to-detect-custom-objects/, 2017. Accessed 10-15-2019.